# Finite-valued Streaming String Transducers

Emmanuel Filiot*
emmanuel.filiot@ulb.be
Université libre de Bruxelles
Bruxelles, Belgium

Ismaël Jecker†
ismael.jecker@gmail.com
Université de Besançon
Besançon, France

Christof Löding
loeding@cs.rwth-aachen.de
RWTH Aachen University
Aachen, Germany

Anca Muscholl
anca@labri.fr
LaBRI, Université Bordeaux
Bordeaux, France

Gabriele Puppis‡
gabriele.puppis@uniud.it
University of Udine
Udine, Italy

Sarah Winter
sarah.winter@irif.fr
IRIF, Université Paris-Cité
Paris, France

## ABSTRACT

A transducer is finite-valued if for some bound $k$, it maps any given input to at most $k$ outputs. For classical, one-way transducers, it is known since the 80s that finite valuedness entails decidability of the equivalence problem. This decidability result is in contrast to the general case, which makes finite-valued transducers very attractive. For classical transducers it is also known that finite valuedness is decidable and that any $k$-valued finite transducer can be decomposed as a union of $k$ single-valued finite transducers.

In this paper, we extend the above results to copyless streaming string transducers (SSTs), answering questions raised by Alur and Deshmukh in 2011. SSTs strictly extend the expressiveness of one-way transducers via additional variables that store partial outputs. We prove that any $k$-valued SST can be effectively decomposed as a union of $k$ (single-valued) deterministic SSTs. As a corollary, we obtain equivalence of SSTs and two-way transducers in the finite-valued case (those two models are incomparable in general). Another corollary is an elementary upper bound for checking equivalence of finite-valued SSTs. The latter problem was already known to be decidable, but the proof complexity was unknown (it relied on Ehrenfeucht's conjecture). Finally, our main result is that finite valuedness of SSTs is decidable. The complexity is PSpace, and even PTime when the number of variables is fixed.

## CCS CONCEPTS

• **Theory of computation → Transducers**; **Streaming models**.

## KEYWORDS

Streaming string transducers, finite valuedness, decomposition

**ACM Reference Format:**
Emmanuel Filiot, Ismaël Jecker, Christof Löding, Anca Muscholl, Gabriele Puppis, and Sarah Winter. 2024. Finite-valued Streaming String Transducers. In *39th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '24), July 8–11, 2024, Tallinn, Estonia.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3661814.3662095

## 1 INTRODUCTION

Finite-state word transducers are simple devices that allow to reason about data transformations in an effective, and even efficient way. In their most basic form they transform words using finite control. Unlike automata, their power heavily depends on various parameters, like non-determinism, the capability of scanning the input several times, or the kind of storage they may use. The oldest transducer model, known as generalized sequential machine, extends finite automata by outputs. Inspired by an approach that applies to arbitrary relational structures [13], logic-based transductions were considered by Engelfriet and Hoogeboom [19] and shown to be equivalent to functional (a.k.a. single-valued) *two-way transducers*. Ten years later Alur and Cerný [2] proposed *streaming string transducers* (SST), a one-way model that uses write-only registers as additional storage. Registers can be updated by appending or prepending strings, or concatenated, but not duplicated (they are copyless). They showed that SSTs are equivalent to the model studied by [19] in the functional case. These equivalences support thus the notion of "regular" word functions, in the spirit of classical results on regular word languages from automata theory and logics due to Büchi, Elgot, Trakhtenbrot, Rabin, and others.

In the non-functional case the picture is less satisfactory, as equivalence only holds between SSTs and non-deterministic MSO transductions [5], which extend the original MSO transductions by existentially quantified monadic parameters. On the other hand, two-way transducers and SSTs become incomparable. Moreover, the equivalence problem for the non-deterministic models is undecidable, even for one-way transducers [22, 28]. There is however a class of relations that is almost as nice as (regular) word functions: this is the class of *finite-valued* relations, namely, relations that associate a uniformly bounded number of outputs with each input.

*Example 1.1.* Let $n \in \mathbb{N}$ and consider the relation

$$R_n = \left\{(\bar{v}_1 \dots \bar{v}_k, \bar{v}_i) \mid k \in \mathbb{N},\ 1 \le i \le k,\ \forall 1 \le j \le k,\ \bar{v}_j \in \{0,1\}^n\right\}.$$

*Emmanuel Filiot is a senior research associate at F.R.S.-FNRS. Research on this work by Emmanuel Filiot was partly funded by the F.R.S.-FNRS – under the MIS project F451019F.
†This work is partly funded by the ERC grant INFSYS (agreement no. 950398) and by the EIPHI Graduate School (contract ANR-17-EURE-0002).
‡Research on this work by Gabriele Puppis was partly funded by the GNCS-INdAM, Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica.


*LICS'24, July 8-–12 2024, Tallinn, Estonia*

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/3661814.3662095

E.g. $(001011, 00), (001011, 10), (001011, 11) \in R_2$. For all $n$, the relation $R_n$ is $2^n$-valued and realizable by a (one-way) finite transducer $T_n$, which uses non-determinism to guess an index $i \in \{1, \dots, k\}$. It keeps on reading its input (in successive rounds of length $n$ to read the bit-vectors $\bar{v}_1, \dots, \bar{v}_{i-1}$) while outputting nothing, until it non-deterministically decides to output $\bar{v}_i$ in $n$ successive steps, and then keeps on reading the remaining $\bar{v}_{i+1}, \dots, \bar{v}_k$ while outputting nothing, and accepts. The transducer $T_n$ has $O(n)$ states and the number of accepting runs per input is unbounded because $k$ is arbitrary and if $\bar{v}_x = \bar{v}_y$ for some $x < y$, then this accounts for two accepting runs and a single output. The ambiguity can be bounded, but at the cost of an exponential blow-up, for instance by making the transducer initially guess a word $\overline{v} \in \{0,1\}^n$ (stored in memory), output it, and then verify $\overline{v} = \overline{v_i}$ for some $1 \leq i \leq k$.

Finite-valued transductions were intensively studied in the setting of one-way transducers. In addition to the fact that finite-valued one-way transducers are exponentially more succinct than their finitely-ambiguous counterpart, as illustrated by Example 1.1, one of the main advantages of this class is the decidability of the equivalence problem [14].[1] Inspired by [14] the equivalence problem for $k$-valued SSTs was shown to be decidable in [35]. However both results rely on the Ehrenfeucht conjecture [1, 23] and therefore provide no elementary upper bounds.

For one-way transducers and for a fixed $k$, being $k$-valued can be checked in PTime [26]. It is also known that every $k$-valued one-way transducer can be effectively decomposed into a union of $k$ unambiguous one-way transducers of exponential size [40, 43, 44]. For both two-way transducers and SSTs, checking $k$-valuedness for fixed $k$ is a PSpace problem (see e.g. [5] for SSTs). Decomposing finite-valued SSTs and deciding finite-valuedness for SSTs were listed as open problems in [5], more than 10 years back. Compared to one-way finite transducers, new challenges arise with SSTs, due to the extra power they enjoy to produce outputs. For example over the alphabet $\Sigma = \{0,1\}$, consider the set of pairs $(w, 0^{n_0}1^{n_1})$ and $(w, 1^{n_1}0^{n_0})$, where $w \in \Sigma^*$ and $n_b$ is the number of bits $b \in \{0,1\}$ in $w$. The outputs are essentially the concatenated numbers of 0 and 1, in unary, and in any order. This transduction is 2-valued, but not realizable by any one-way finite transducer. However, the following single-state SST $T$ with 2 registers $X_0, X_1$ (both initially empty) realizes it: whenever it reads $b \in \{0,1\}$, it non-deterministically does the update $X_b := bX_b$ or $X_b := X_b b$ (while $X_{1-b}$ is unchanged), and eventually outputs either $X_0X_1$ or $X_1X_0$. The ability of SSTs to generate outputs non-linearly makes their study challenging. To illustrate this, consider the following slight modification of $T$: $X_0$ is initialized with 1 instead of the empty word. The resulting SST is not finite-valued anymore, because on reading $0^n$, it would output $0^\ell 10^m$ for all $\ell, m$ such that $\ell + m = n$.

Another open problem was the relationship between SSTs and two-way transducers in the finite-valued case. It is not hard to see that the standard translation from deterministic two-way transducers to deterministic SSTs also applies to the finite-valued case, see the proof of Theorem 1.3 below. The converse translation however is far more complicated and relies on a decomposition theorem for SSTs, which we establish here.

[1] [14] states that their proof works for two-way transducers as well.

**Contributions.** The results presented in this paper draw a rather complete picture about finite-valued SSTs, answering several open problems from [5]. First, we show that $k$-valued SSTs enjoy the same decomposition property as one-way transducers:

Theorem 1.2. *For all $k \in \mathbb{N}$, every $k$-valued SST can be decomposed into a union of $k$ single-valued (or even deterministic) SSTs. The complexity of the construction is elementary.*

An important consequence of the above theorem is the equivalence of SSTs and two-way transducers in the finite-valued setting:

Theorem 1.3. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a finite-valued relation. If $R$ can be realized by an SST, then an equivalent two-way transducer can be effectively constructed, and vice-versa.*

A second consequence of Theorem 1.2 is an elementary upper bound for the equivalence problem of finite-valued SSTs [35]:

Theorem 1.4. *The equivalence problem for $k$-valued SSTs can be solved with elementary complexity.*

Proof. Given two $k$-valued SSTs $T, T'$, we first decompose them into unions of $k$ single-valued SSTs $T_1, \dots, T_k$ and $T'_1, \dots, T'_k$, respectively. By [2] we can even assume that every $T_i$ and $T'_j$ is a deterministic SST. Finally, [5, Theorem 4.4] shows how to check the equivalence of $\bigcup_{i=1}^k T_i$ and $\bigcup_{i=1}^k T'_i$ in PSpace. □

Our last, and main, contribution is to establish the decidability of finite valuedness for SSTs:

Theorem 1.5. *Given any SST $T$, we can decide in* PSpace if $T$ is finite-valued, and in PTime if the number of variables is fixed. Moreover, this problem is at least as hard as the equivalence problem for deterministic SSTs.

This last result is the most technical one, and requires to reason on particular substructures (W-patterns) of SSTs. Such substructures have been already used for one-way transducers, but for SSTs genuine challenges arise. The starting point of our proof is a recent result allowing to determine if two runs of an SST are far apart [20]. The proof then relies on identifying suitable patterns and extending techniques from word combinatorics to more involved word inequalities.

Based on the equivalence between SSTs and two-way transducers in the finite-valued setting (Theorem 1.3), and the decidability of finite valuedness for SST (Theorem 1.5), we exhibit an alternative proof for the following (known) result:

Corollary 1.6 ([46]). *Finite-valuedness of two-way transducers is decidable in* PSpace.

Observe also that without the results in this paper, the result of [46] could not help to show Theorem 1.5, because only the conversion of finite-valued two-way transducers into finite-valued SSTs was known (under the assumption that any input positions is visited a bounded number of times), but not the other way around. Finally note that Theorem 1.2 (and Theorem 1.3) also provides a decomposition result for finite-valued two-way transducers.

We also observe that since SSTs and non-deterministic MSO transductions are equivalent [5], Theorem 1.5 entails decidability of finite valuedness for non-deterministic MSO transductions as well.

Moreover, since in the single-valued case, deterministic SSTs and MSO transductions are also equivalent [2], Theorem 1.2 implies a decomposition result for MSO transductions: any $k$-valued nondeterministic MSO transduction can be decomposed as a union of $k$ (deterministic) MSO transductions. Finally, from Theorem 1.3, we also get that under the assumption of finite valuedness, nondeterministic MSO transductions, two-way transducers and SST are equally expressive.

A long version of the paper is available at http://arxiv.org/abs/2405.08171.

## 2 PRELIMINARIES

For convenience, technical terms and notations in the electronic version of this manuscript are hyper-linked to their definitions (cf. https://ctan.org/pkg/knowledge).

Hereafter, $\mathbb{N}$ (resp. $\mathbb{N}_+$) denotes the set of non-negative (resp. strictly positive) integers, and $\Sigma$ denotes a generic alphabet.

**Words and relations.** We denote by $\varepsilon$ the empty word, by $|u|$ the length of a word $u \in \Sigma^*$, and by $u[i]$ its $i$-th letter, for $1 \le i \le |u|$. We introduce a convolution operation on words, which is particularly useful to identify robust and well-behaved classes of relations, as it is done for instance in the theory of automatic structures [8]. For simplicity, we only consider convolutions of words of the same length. Given $u, v \in \Sigma^*$, with $|u| = |v|$, the *convolution* $u \otimes v$ is a word over $(\Sigma^2)^*$ of length $|u| = |v|$ such that $(u \otimes v)[i] = (u[i], v[i])$ for all $1 \le i \le |u|$. For example, $(aba) \otimes (bcc) = (a,b)(b,c)(a,c)$. As $\otimes$ is associative, we may write $u \otimes v \otimes w$ for any words $u, v, w$.

A relation $R \subseteq (\Sigma^*)^k$ is *length-preserving* if $|u_1| = \cdots = |u_k|$ for all $(u_1, \ldots, u_k) \in R$. A length-preserving relation is *automatic* if the language $\{u_1 \otimes \ldots \otimes u_k \mid (u_1, \ldots, u_k) \in R\}$ is recognized by a finite state automaton. A binary relation $R \subseteq \Sigma^* \times \Sigma^*$ (not necessarily length-preserving) is *$k$-valued*, for $k \in \mathbb{N}$, if for all $u \in \Sigma^*$, there are at most $k$ words $v$ such that $(u, v) \in R$. It is *finite-valued* if it is $k$-valued for some $k$.

**Variable updates.** Fix a finite set of variables $\mathcal{X} = \{X_1, \ldots, X_m\}$, disjoint from the alphabet $\Sigma$. A (copyless) *update* is any mapping $\alpha : \mathcal{X} \to (\Sigma \uplus \mathcal{X})^*$ such that each variable $X \in \mathcal{X}$ appears *at most once* in the word $\alpha(X_1) \ldots \alpha(X_m)$. Such an update can be morphically extended to words over $\Sigma \uplus \mathcal{X}$, by simply letting $\alpha(a) = a$ for all $a \in \Sigma$. Using this, we can compose any two updates $\alpha, \beta$ to form a new update $\alpha\,\beta : \mathcal{X} \to (\Sigma \uplus \mathcal{X})^*$, defined by $(\alpha\,\beta)(X) = \beta(\alpha(X))$ for all $X \in \mathcal{X}$. An update is called *initial* (resp. *final*) if all variables in $\mathcal{X}$ (resp. $\mathcal{X} \setminus \{X_1\}$) are mapped to the empty word. The designated variable $X_1$ is used to store the final output produced by an SST, as defined in the next paragraph.

**Streaming string transducers.** A (non-deterministic, copyless) *streaming string transducer* (*SST* for short) is a tuple $T = (\Sigma, \mathcal{X}, Q, Q_{\text{init}}, Q_{\text{final}}, \Omega, \Delta)$, where $\Sigma$ is an alphabet, $Q$ is a finite set of states, $Q_{\text{init}}, Q_{\text{final}} \subseteq Q$ are the sets of initial and final states, $\Omega$ is a function from final states to final updates, and $\Delta$ is a finite transition relation consisting of tuples of the form $(q, a, \alpha, q')$, where $q, q' \in Q$ are the source and target states, $a \in \Sigma$ is an input symbol, and $\alpha$ is an update. We often denote a transition $(q, a, \alpha, q') \in \Delta$ by the annotated arrow:

$$q \xrightarrow{a/\alpha} q'.$$

The *size* $|T|$ of an SST $T$ is defined as the number of states plus the size of its transition relation.

A *run* of $T$ is a sequence of transitions from $\Delta$ of the form

$$\rho = q_0 \xrightarrow{a_1/\alpha_1} q_1 \xrightarrow{a_2/\alpha_2} q_2 \cdots q_{n-1} \xrightarrow{a_n/\alpha_n} q_n.$$

The *input consumed by* $\rho$ is the word $\text{in}(\rho) = a_1 \ldots a_n$. The *update induced by* $\rho$ is the composition $\beta = \alpha_1 \ldots \alpha_n$. We write $\rho : u/\beta$ to mean that $\rho$ is a run with $u$ as consumed input and $\beta$ as induced update. A run $\rho$ as above is *accepting* if the first state is initial and the last state is final, namely, if $q_0 \in Q_{\text{init}}$ and $q_n \in Q_{\text{final}}$. In this case, the induced update, extended to the left with the initial update denoted by $\iota$ and to the right with the final update $\Omega(q_n)$, gives rise to an update $\iota\,\beta\,\Omega(q_n)$ that maps $X_1$ to a word over $\Sigma$ and all remaining variables to the empty word. In particular, the latter update determines the *output produced by* $\rho$, defined as the word $\text{out}(\rho) = (\iota\,\beta\,\Omega(q_n))(X_1)$.

The *relation realized by* an SST $T$ is

$$\mathscr{R}(T) \;=\; \big\{\big(\text{in}(\rho), \text{out}(\rho)\big) \in \Sigma^* \times \Sigma^* \;\big|\; \rho \text{ accepting run of } T\big\}$$

An SST is *$k$-valued* (resp. *finite-valued*) if its realized relation is so. It is *deterministic* if it has a single initial state and the transition relation is a partial function (from pairs of states and input letters to pairs of updates and states). It is *unambiguous* if it admits at most one accepting run on each input. Similarly, it is called *$k$-ambiguous* if it admits at most $k$ accepting runs on each input. Of course, every deterministic SST is unambiguous, and every unambiguous SST is single-valued (i.e. 1-valued). Two SSTs $T_1, T_2$ are *equivalent* if $\mathscr{R}(T_1) = \mathscr{R}(T_2)$. The equivalence problem for SSTs is undecidable in general, and it is so even for one-way transducers [22, 28]. However, decidability is recovered for finite-valued SSTs:

Theorem 2.1 ([35]). *Equivalence of finite-valued SSTs is decidable.*

Note that checking equivalence is known to be in PSpace for *deterministic* SSTs. This easily generalizes to unions of deterministic (hence single-valued) SSTs, because the equivalence checking algorithm is exponential only in the number of variables:

Theorem 2.2 ([5]). *The following problem is in* PSpace: given $n + m$ deterministic SSTs $T_1, \ldots, T_n, T'_1, \ldots, T'_m$, decide whether $\bigcup_{i=1}^{n} \mathscr{R}(T_i) = \bigcup_{j=1}^{m} \mathscr{R}(T'_j)$.

For any fixed $k$, the $k$-valuedness property is decidable in PSpace:

Theorem 2.3 ([5]). *For any fixed $k \in \mathbb{N}$, the following problem is in*[2] PSpace: given an SST $T$, decide whether $T$ is $k$-valued. It is in PTime if one further restricts to SSTs with a fixed number of variables.

The decidability status of finite valuedness for SSTs, that is, for unknown $k$, was an open problem. Part of our contribution is to show that this problem is also decidable.

[2] In [5], no complexity result is provided, but the decidability procedure relies on a reduction to the emptiness of a 1-reversal $k(k+1)$-counter machine, based on the proof for equivalence of deterministic SST [3]. The counter machine is exponential in the number of variables only, and the result follows since emptiness of counter machines with fixed number of reversals and fixed number of counters is in NLogSpace [24].

## 2.1 Pumping and word combinatorics

When reasoning with automata, it is common practice to use pumping arguments. This section introduces pumping for SSTs, as well as combinatorial results for reasoning about (in)equalities between pumped outputs of SSTs.

In order to have adequate properties for pumped runs of SSTs, the notion of loop needs to be defined so as to take into account how the content of variables "flows" into other variables when performing an update. We define the *skeleton* of an update $\alpha : \mathcal{X} \to (\Sigma \uplus \mathcal{X})^*$ as the update $\hat{\alpha} : \mathcal{X} \to \mathcal{X}^*$ obtained from $\alpha$ by removing all the letters from $\Sigma$. Note that there are only finitely many skeletons, and their composition forms a finite monoid, called the *skeleton monoid* (this is very similar to the flow monoid from [35], but does not rely on any normalization).

A *loop* of a run $\rho$ of an SST is any factor $L$ of $\rho$ that starts and ends in the same state and induces a *skeleton-idempotent* update, namely, an update $\alpha$ such that $\alpha$ and $\alpha\,\alpha$ have the same skeleton. For example, the update $\alpha : X_1 \mapsto a\,X_1\,b\,X_2\,c,\ X_2 \mapsto a$ is skeleton-idempotent and thus can be part of a loop. We will often denote a loop in a run by an interval $[i, j]$. In this case, it is convenient to assume that the indices $i, j$ represent "positions" in-between the transitions, thus identifying occurrences of states; in this way, adjacent loops can be denoted by intervals of the form $[i_1, i_2]$, $[i_2, i_3]$, etc. In particular, if the run consists of $n$ transitions, then the largest possible interval on it is $[0, n]$. For technical reasons, we do allow *empty* loops, that is, loops of the form $[i, j]$, with $i = j$ and with the induced update being the identity function on $\mathcal{X}$.

The run obtained from $\rho$ by pumping $n$ times a loop $L$ is denoted $\mathrm{pump}_L^n(\rho)$. If we are given an $m$-tuple of *pairwise disjoint* loops $\bar{L} = (L_1, \ldots, L_m)$ and an $m$-tuple of (positive) numbers $\bar{n} = (n_1, \ldots, n_m)$, then we write $\mathrm{pump}_{\bar{L}}^{\bar{n}}(\rho)$ for the run obtained by pumping simultaneously $n_i$ times $L_i$, for each $1 \le i \le m$.

The next lemma is a Ramsey-type argument that, based on the number of states of the SST, the size of the skeleton monoid, and a number $n$, derives a minimum length for a run to witness $n + 1$ points and loops between pairs of any of these points. The reader can refer to [30] to get good estimates of the values of $E, H$.

Lemma 2.4. *Given an SST, one can compute two numbers $E, H$ such that for every run $\rho$, every $n \in \mathbb{N}$, and every set $I \subseteq \{0, \ldots, |\rho|\}$ of cardinality $En^H + 1$, there is a subset $I' \subseteq I$ of cardinality $n + 1$ such that for all $i < j \in I'$ the interval $[i, j]$ is a loop of $\rho$. The values of $E, H$ are elementary in the size of the SST.*

Below, we describe the effect on the output of pumping loops in a run of an SST. We start with the following simple combinatorial result:

Lemma 2.5. *Let $\alpha$ be a skeleton-idempotent update. For every variable $X$, there exist two words $u, v \in \Sigma^*$ such that, for all positive natural numbers $n \in \mathbb{N}_+$, $\alpha^n(X) = u^{n-1}\,\alpha(X)\,v^{n-1}$.*

It follows that pumping loops in a run corresponds to introducing repeated copies of factors in the output. Similar results can be found in [35] for SSTs and in [18, 36] for two-way transducers:

Corollary 2.6. *Let $\rho$ be an accepting run of an SST and let $\bar{L} = (L_1, \ldots, L_m)$ be a tuple of pairwise disjoint loops in $\rho$. Then, for some $r \le 2m|\mathcal{X}|$ there exist words $w_0, \ldots, w_r, u_1, \ldots, u_r$ and indices $1 \le i_1, \ldots, i_r \le m$, not necessarily distinct, such that for every tuple $\bar{n} = (n_1, \ldots, n_m) \in \mathbb{N}_+^m$ of positive natural numbers,*

$$\mathrm{out}(\mathrm{pump}_{\bar{L}}^{\bar{n}}(\rho)) \;=\; w_0\,u_1^{n_{i_1}-1}\,w_1\,\ldots\,u_r^{n_{i_r}-1}\,w_r.$$

Proof. This follows immediately from Lemma 2.5. Note that the content of any variable $X$ just after pumping a loop either appears as infix of the final output, or is erased by some later update. In both cases, each pumped loop $L_i$ induces in the output $(n_i - 1)$-folded repetitions of $2k$ (possibly empty) factors, where $k$ is the number of variables of the SST. Since the loops are pairwise disjoint, they contribute such factors without any interference. The final output $\mathrm{out}(\mathrm{pump}_{\bar{L}}^{\bar{n}}(\rho))$ thus features repetitions of $r = 2km$ (possibly empty) factors. □

The rest of the section analyses properties of words with repeated factors like the one in Corollary 2.6.

*Definition 2.7.* A *word inequality* with repetitions parametrized in $\mathtt{X}$ is a pair $e = (w, w')$ of terms of the form

$$w \;=\; s_0\,t_1^{\mathtt{x}_1}\,s_1\,\ldots\,t_m^{\mathtt{x}_m}\,s_m$$
$$w' \;=\; s_0'\,t_1'^{\mathtt{y}_1}\,s_1'\,\ldots\,t_n'^{\mathtt{y}_n}\,s_n'$$

where $s_i, t_i, s_j', t_j' \in \Sigma^*$ and $\mathtt{x}_i, \mathtt{y}_j \in \mathtt{X}$ for all $i, j$. The set of *solutions* of $e = (w, w')$, denoted $\mathrm{Sols}(e)$, consists of the mappings $f : \mathtt{X} \to \mathbb{N}$ such that $f(w) \neq f(w')$, where $f(w)$ is the word obtained from $w$ by substituting every formal parameter $\mathtt{x} \in \mathtt{X}$ by $f(\mathtt{x})$, and similarly for $f(w')$. A *system of word inequalities* is a non-empty finite set $E$ of inequalities as above, and its set of solutions is given by $\mathrm{Sols}(E) = \bigcap_{e \in E} \mathrm{Sols}(e)$.

The next theorem states that if there exists a solution to a system of inequalities parameterized by a single variable $\mathtt{x}$, then the set of solutions is co-finite.

Theorem 2.8 ([37, Theorem 4.3]). *Given a word inequality $e$ with repetitions parameterized by single variable $\mathtt{x}$, $\mathrm{Sols}(e)$ is either empty or co-finite; more precisely, if the left (resp. right) hand-side of $e$ contains $m$ (resp. $n$) repeating patterns (as in Definition 2.7), then either $\mathrm{Sols}(e) = \emptyset$ or $|\mathbb{N} \setminus \mathrm{Sols}(e)| \le m + n$.*

Finally, we present two corollaries of the above theorem, that will be used later. The first corollary concerns satisfiability of a system of inequalities. Formally, we say that a word inequality $e$ (resp. a system of inequalities $E$) is *satisfiable* if its set of solutions is non-empty.

Corollary 2.9. *Let $E$ be a system of word inequalities. If every inequality $e \in E$ is satisfiable, then so is the system $E$.*

The second corollary is related to the existence of large sets of solutions for a satisfiable word inequality that avoid any correlation between variables. To formalize the statement, it is convenient to fix a total order on the variables of the inequality, say $\mathtt{x}_1, \ldots, \mathtt{x}_k$, and then identify every function $f : \mathtt{X} \to \mathbb{N}$ with the $k$-tuple of values $\bar{x} = (x_1, \ldots, x_k)$, where $x_i = f(\mathtt{x}_i)$ for all $i = 1, \ldots, k$. According to this correspondence, the corollary states the existence of sets of solutions that look like Cartesian products of finite intervals of values, each with arbitrarily large cardinality. The statement of the corollary is in fact slightly more complicated than this, as it discloses dependencies between the intervals. We also observe that

the order in which we list the variables is arbitrary, but different orders will induce different dependencies between intervals.

Corollary 2.10. *Let $e$ be a word inequality with repetitions parametrized in $\mathsf{X} = \{x_1, \ldots, x_k\}$. If $e$ is satisfiable, then*

$$\exists \ell_1 \, \forall h_1 \; \ldots \; \exists \ell_k \, \forall h_k \quad \underbrace{[\ell_1, h_1]}_{\text{values for } x_1} \times \ldots \times \underbrace{[\ell_k, h_k]}_{\text{values for } x_k} \subseteq \mathrm{Sols}(e).$$

## 2.2 Delay between accepting runs

We briefly recall the definitions from [20] and introduce a measure of similarity (called delay) between accepting runs of an SST that have the same input and the same output.

We first give some intuition, followed by definitions and an example. Naturally, the difference between the amount of output symbols produced during a run should be an indicator of (dis)similarity. However, as SSTs do not necessarily build their output from left to right, one must also take into account the position where an output symbol is placed. For example, compare two runs $\rho$ and $\rho'$ on the same input that produce the same output $aaabbb$. After consuming a prefix of the input, $\rho$ may have produced $aaa\_\_\_$ and $\rho'$ may have produced $\_\_\_bbb$. The amount of produced output symbols is the same, but the runs are delayed because $\rho$ built the output from the left, whereas $\rho'$ did it from the right. This idea of delay comes with an important caveat. As another example, consider two runs $\rho$ and $\rho'$ on the same input that produce the same output $aaaaaa$, and assume that, after consuming the same prefix of the input, $\rho$ and $\rho'$ produced $aaa\_\_\_$ and $\_\_\_aaa$, respectively. Note that the output $aaaaaa$ is a periodic word. Hence, it does not matter if $aaa$ is appended or prepended to a word with period $a$. In general, one copes with this phenomenon by dividing the output into periodic parts, where all periods are bounded by a well-chosen parameter $C$. So, intuitively, the delay measures the difference between the numbers of output symbols that have been produced by the two runs, up to the end of each of periodic factor. The number of produced output symbols is formally captured by a weight function, defined below, and the delay aggregates the weight differences.

For an accepting run $\rho$, a position $t$ of $\rho$, and a position $j$ in the output $\mathrm{out}(\rho)$, we denote by $\mathrm{weight}_j^t(\rho)$ the number of output positions $j' \leq j$ that are produced by the prefix of $\rho$ up to position $t$. We use the above notation when $j$ witnesses a change in a repeating pattern of the output. These changes in repeating patterns are called cuts, as formalized below.

Let $w$ be any non-empty word (e.g. the output of $\rho$ or a factor of it). The *primitive root* of $w$, denoted $\mathrm{root}(w)$, is the shortest word $r$ such that $w \in \{r\}^*$. For a fixed integer $C > 0$ we define a factorization $w[1, j_1], w[j_1+1, j_2], \ldots, w[j_n+1, j_{n+1}]$ of $w$ in which every $j_i$ is chosen as the rightmost position for which $w[j_{i-1} + 1, j_i]$ has primitive root of length not exceeding $C$. These positions $j_1, \ldots, j_n$ are called *$C$-cuts*. More precisely:

- the *first $C$-cut* of $w$ is the largest position $j \leq |w|$, such that $|\mathrm{root}(w[1, j])| \leq C$;
- if $j$ is the $i$-th $C$-cut of $w$, then the *$(i+1)$-th $C$-cut* of $w$ is the largest position $j' > j$ such that $|\mathrm{root}(w[j+1, j'])| \leq C$.

We denote by $C\text{-cuts}(w)$ the set of all $C$-cuts of $w$.

We are now ready to define the notion of delay. Consider two accepting runs $\rho, \rho'$ of an SST with the same input $u = \mathrm{in}(\rho) = \mathrm{in}(\rho')$ and the same output $w = \mathrm{out}(\rho) = \mathrm{out}(\rho')$, and define:

$$C\text{-delay}(\rho, \rho') = \max_{\substack{t \leq |u|, \\ j \in C\text{-cuts}(w)}} \left|\mathrm{weight}_j^t(\rho) - \mathrm{weight}_j^t(\rho')\right|$$

that is, the maximum, over all prefixes of the runs and all $C$-cuts of the output, of the absolute values of the differences of the corresponding weights in $\rho$ and $\rho'$. Note that the delay is only defined for accepting runs with same input and output. So whenever we write $C\text{-delay}(\rho, \rho')$, we implicitly mean that $\rho, \rho'$ have *same input and same output*.

*Example 2.11.* Let $w = abcccbb$ be the output of runs $\rho, \rho'$ on the same input of length 2. Assume $\rho$ produces $abc\_\_bb$ and then $abcccbb$, whereas $\rho'$ produces $\_\_\_c\_bb$ and then $abcccbb$. For $C = 2$, we obtain $2\text{-cuts}(w) = \{2, 5, 7\}$, i.e., $w$ is divided into $ab|ccc|bb$. To compute the $2\text{-delay}(\rho, \rho')$, we need to calculate weights at cuts. For $t = 0$, $\mathrm{weight}_j^0(\rho) = \mathrm{weight}_j^0(\rho') = 0$ for all $j \in 2\text{-cuts}(w)$ because nothing has been produced. For $t = 2$, $\mathrm{weight}_j^2(\rho) = \mathrm{weight}_j^2(\rho') = j$ for all $j \in 2\text{-cuts}(w)$ because the whole output has been produced. Only the case $t = 1$ has an impact on the delay. We have $\mathrm{weight}_2^1(\rho) = 2$, $\mathrm{weight}_5^1(\rho) = 3$, and $\mathrm{weight}_7^1(\rho) = 5$. Also, we have $\mathrm{weight}_2^1(\rho') = 0$, $\mathrm{weight}_5^1(\rho) = 1$, and $\mathrm{weight}_7^1(\rho) = 3$. Hence, we obtain $2\text{-delay}(\rho, \rho') = 2$.

We recall below a few crucial results from [20]. A first result shows that the relation of pairs of runs having bounded delay (for a fixed bound) is automatic — for this to make sense, we view a run of an SST as a finite word, with letters representing transitions, and we recall that a relation is automatic if its convolution language is regular.

Lemma 2.12 ([20, Theorem 5]). *Given an SST and some numbers $C, D$, the relation consisting of pairs of accepting runs $(\rho, \rho')$ such that $C\text{-delay}(\rho, \rho') \leq D$ is automatic.*

Proof. The statement in [20, Theorem 5] is not for runs of SSTs, but for sequences of updates. One can easily build an automaton that checks if two sequences of transitions, encoded by their convolution, form accepting runs $\rho, \rho'$ of the given SST on the same input. The remaining condition $C\text{-delay}(\rho, \rho') \leq D$ only depends on the underlying sequences of updates determined by $\rho$ and $\rho'$, and can be checked using [20, Theorem 5]. □

A second result shows that given two runs with large delay, one can find a set of positions on the input (the cardinality of which depends on how large the delay is) in such a way that any interval starting just before any of these positions and ending just after any other of these positions is a loop on both runs such that, when pumped, produces different outputs. By this last result, large delay intuitively means "potentially different outputs".

Lemma 2.13 ([20, Lemma 6]). *Given an SST, one can compute[3] some numbers $C, D$ such that, for all $m \geq 1$ and all runs $\rho, \rho'$: if*

[3] We remark that the notation and the actual bounds here differ from the original presentation of [20], mainly due to the fact that here we manipulate runs with explicit states and loops with idempotent skeletons. In particular, the parameters $C, D, m$ here correspond respectively to the values $kE^2, \ell E^4, CE^2$ with $k, \ell, C$ as in [20, Lemma 6], and $E$ as in our Lemma 2.4.

*$C$m-delay$(\rho, \rho') > Dm^2$, then there exist $m$ positions $0 \le \ell_1 < \cdots < \ell_m \le |\rho|$ such that, for every $1 \le i < j \le m$, the interval $L_{i,j} = [\ell_i, \ell_j]$ is a loop on both $\rho$ and $\rho'$ and satisfies*

$$\text{out}(\text{pump}^2_{L_{i,j}}(\rho)) \neq \text{out}(\text{pump}^2_{L_{i,j}}(\rho')).$$

To reason about finite valuedness we will often consider several accepting runs on the same input that have pairwise large delays. By Lemma 2.13, every two such runs can be pumped so as to witness different outputs. The crux however is to show that these runs can be pumped *simultaneously* so as to get pairwise different outputs. This is indeed possible thanks to:

Lemma 2.14. *Let $C, D$ be computed as in Lemma 2.13, and $k$ be an arbitrary number. Then one can compute a number $m$ such that, for all runs $\rho_0, \ldots, \rho_k$ on the same input and with $\bigwedge_{0 \le i < j \le k} \big(\text{out}(\rho_i) \neq \text{out}(\rho_j) \ \vee \ C\text{m-delay}(\rho_i, \rho_j) > Dm^2\big)$, there is a tuple $\bar{L} = (L_{i,j})_{0 \le i < j \le k}$ of disjoint intervals that are loops on all runs $\rho_0, \ldots, \rho_k$, and there is a tuple $\bar{n} = (n_{i,j})_{0 \le i < j \le k}$ of positive numbers such that*

$$\text{for all } 0 \le i < j \le k, \quad \text{out}(\text{pump}^{\bar{n}}_{\bar{L}}(\rho_i)) \neq \text{out}(\text{pump}^{\bar{n}}_{\bar{L}}(\rho_j)).$$

Proof. We first define $m$. Let $E, H$ be as in Lemma 2.4, and set $m := m_0 + 1$ for the sequence $m_0, \ldots, m_{k-1}$ defined inductively by $m_{k-1} := k(k+1)$, and $m_h := Em^H_{h+1}$.

We show how to pump the runs in such a way that all pairs of indices $i < j$ witnessing $C$m-delay$(\rho_i, \rho_j) > Dm^2$ before pumping, will witness different outputs after pumping. Consider one such pair $(i, j)$, with $i < j$, such that $C$m-delay$(\rho_i, \rho_j) > Dm^2$, so in particular, $\text{out}(\rho_i) = \text{out}(\rho_j)$ (if there is no such pair, then all runs have pairwise different outputs, and so we are already done). We apply Lemma 2.13 and obtain a set $I_{i,j,0}$ of $m = m_0 + 1$ positions such that each interval $L = [\ell, \ell']$ with $\ell, \ell' \in I_{i,j,0}$ is a loop on both $\rho_i$ and $\rho_j$, and:

$$\text{out}(\text{pump}^2_L(\rho_i)) \neq \text{out}(\text{pump}^2_L(\rho_j)). \tag{1}$$

Then, by repeatedly using Lemma 2.4, we derive the existence of sets $I_{i,j,k-1} \subseteq \cdots \subseteq I_{i,j,1} \subseteq I_{i,j,0}$ with $|I_{i,j,h}| = m_h + 1$ such that each interval $L = [\ell, \ell']$ with $\ell, \ell' \in I_{i,j,h}$ is a loop on $\rho_i, \rho_j$, and $h$ further runs from $\rho_0, \ldots, \rho_k$ (our definition of $m$ from the beginning of the proof is tailored to this repeated application of Lemma 2.4, because $|I_{i,j,h}| = m_h + 1 = Em^H_{h+1} + 1$). In particular, all intervals with endpoints in $I_{i,j,k-1}$ are loops on all the $\rho_0, \ldots, \rho_k$.

In this way, for each pair $i < j$ such that $\rho_i$ and $\rho_j$ have large delay, we obtain $k(k+1)$ adjacent intervals that are loops on all runs and that satisfy the pumping property (1) from above.

As there are at most $k(k+1)$ pairs of runs, we can now choose from the sets of intervals that we have prepared one interval $L_{i,j}$ for each pair $i < j$ with $C$m-delay$(\rho_i, \rho_j) > Dm^2$, in such a way that all the chosen intervals are pairwise disjoint (for example, we could do so by always picking among the remaining intervals the one with the left-most right border, and then removing all intervals that intersect this one). The selected intervals $L_{i,j}$ thus have the following properties:

(1) $L_{i,j}$ is a loop on all runs $\rho_0, \ldots, \rho_k$,
(2) $L_{i,j}$ is disjoint from every other interval $L_{i',j'}$,
(3) $\text{out}(\text{pump}^2_{L_{i,j}}(\rho_i)) \neq \text{out}(\text{pump}^2_{L_{i,j}}(\rho_j))$.

If a pair $i < j$ of runs is such that $\text{out}(\rho_i) \neq \text{out}(\rho_j)$, then we set $L_{i,j}$ as an empty loop.

Now, let $\bar{L} = (L_{i,j})_{0 \le i < j \le k}$ be the tuple of chosen intervals, and consider the following system of word inequalities with formal parameters $(\mathsf{x}_{i,j})_{0 \le i < j \le k} =: \bar{\mathsf{x}}$:

$$\text{for all } 0 \le i < j \le k, \quad \text{out}(\text{pump}^{\bar{\mathsf{x}}}_{\bar{L}}(\rho_i)) \neq \text{out}(\text{pump}^{\bar{\mathsf{x}}}_{\bar{L}}(\rho_j)).$$

Here, the value of the formal parameter $\mathsf{x}_{i,j}$ determines how often the loop $L_{i,j}$ is pumped. By Corollary 2.6, this corresponds to a word inequality in the parameters $\mathsf{x}_{i,j}$.

Note that there is one such inequality for each pair of runs $\rho_i, \rho_j$ with $0 \le i < j \le k$. By the choice of the intervals in $\bar{L}$, each of the inequalities is satisfiable: indeed, the inequality for $\rho_i, \rho_j$ is satisfied by letting $\mathsf{x}_{i',j'} = 1$ if $i' \neq i$ or $j' \neq j$, and $\mathsf{x}_{i,j} = 2$ otherwise.

By Corollary 2.9, the system of inequalities is also satisfiable with a tuple $\bar{n} = (n_{i,j})_{0 \le i < j \le k}$ of numbers, as claimed in the lemma. □

## 3 THE DECOMPOSITION THEOREM

This section is devoted to the proof of the Decomposition Theorem:

Theorem 1.2. *For all $k \in \mathbb{N}$, every $k$-valued SST can be decomposed into a union of $k$ single-valued (or even deterministic) SSTs. The complexity of the construction is elementary.*

Our proof relies on the notion of *cover* of an SST, which is reminiscent of the so-called "lag-separation covering" construction [17, 39]. Intuitively, given an SST $T$ and two integers $C, D \in \mathbb{N}$, we construct an SST $\text{Cover}_{C,D}(T)$ that is equivalent to $T$, yet for each input $u$ it only admits pairs of accepting runs with different outputs or $C$-delay larger than $D$.

Proposition 3.1. *Given an SST $T$ and two numbers $C, D$, one can compute an SST called $\text{Cover}_{C,D}(T)$ such that*

*(1) $\text{Cover}_{C,D}(T)$ is equivalent to $T$;*
*(2) for every two accepting runs $\rho \neq \rho'$ of $\text{Cover}_{C,D}(T)$ having the same input, either $\text{out}(\rho) \neq \text{out}(\rho')$ or $C$-delay$(\rho, \rho') > D$;*
*(3) every accepting run of $\text{Cover}_{C,D}(T)$ can be projected onto an accepting run of $T$.*

Proof. We order the set of accepting runs of $T$ lexicographically, and we get rid of all the runs for which there exists a lexicographically smaller run with the same input, the same output, and small delay. Since all these conditions are encoded by regular languages, the remaining set of runs is also regular, and this can be used to construct an SST $\text{Cover}_{C,D}(T)$ that satisfies the required properties.

We now give more details about this construction. Let $R$ denote the set of all accepting runs of $T$. Remark that $R$ is a language over the alphabet consisting of transitions of $T$, and it is recognised by the underlying automaton of $T$, so it is regular. Let

$$\text{Sep}_{C,D}(R) = \big\{\rho \in R \,\big|\, \nexists \rho' \in R \,.\, \rho' < \rho \ \wedge \ C\text{-delay}(\rho, \rho') \le D\big\}.$$

Recall that the delay is only defined for accepting runs with same input and same output, so $C$-delay$(\rho, \rho') \le D$ implies that $\text{in}(\rho) = \text{in}(\rho')$ and $\text{out}(\rho) = \text{out}(\rho')$. We show that

A) $\text{Sep}_{C,D}(R)$ is a regular subset of $R$;
B) $\{(\text{in}(\rho), \text{out}(\rho)) \mid \rho \in \text{Sep}_{C,D}(R)\} = \{(\text{in}(\rho), \text{out}(\rho)) \mid \rho \in R\}$;
C) for every pair of runs $\rho, \rho' \in \text{Sep}_{C,D}(R)$ over the same input, either $\text{out}(\rho) \neq \text{out}(\rho')$ or $C$-delay$(\rho, \rho') > D$.

Before proving these properties, let us show how to use them to conclude the proof of the proposition:

We start with a DFA $A$ recognizing $\mathrm{Sep}_{C,D}(R)$, whose existence is guaranteed by Property A). Note that the transitions of $A$ are of the form $\big(q, (s, a, \alpha, s'), q'\big)$, where $(s, a, \alpha, s')$ is a transition of $T$. Without loss of generality, we assume that the source state $q$ of an $A$-transition determines the source state $s$ of the corresponding $T$-transition, and similarly for the target states $q'$ and $s'$. Thanks to this, we can turn $A$ into the desired SST $\mathrm{Cover}_{C,D}(T)$ by simply projecting away the $T$-states from the $T$-transitions, namely, by replacing every transition $\big(q, (s, a, \alpha, s'), q'\big)$ with $(q, a, \alpha, q')$. To complete the construction, we observe that if the state $q'$ is final in $A$, then the corresponding state $s'$ is also final in $T$ (this is because $A$ recognizes only accepting runs of $T$). Accordingly, we can define the final update of $\mathrm{Cover}_{C,D}(T)$ so that it maps any final state $q'$ of $A$ to the final update $\Omega(s')$, as determined by the corresponding final state $s'$ in $T$. Finally, thanks to Properties B) and C), the SST $\mathrm{Cover}_{C,D}(T)$ constructed in this way clearly satisfies the properties claimed in the proposition.

Let us now prove Properties A)–C).

Proof of Property A). Note that $\mathrm{Sep}_{C,D}(R)$ is obtained by combining the relations $R$, $\{(\rho, \rho') \mid \rho' < \rho\}$, and $\{(\rho, \rho') \mid C\text{-delay}(\rho, \rho') \leq D\}$ using the operations of intersection, projection, and complement. Also recall that $R$ can be regarded a regular language, and that $\{(\rho, \rho') \mid \rho' < \rho\}$ and $\{(\rho, \rho') \mid C\text{-delay}(\rho, \rho') \leq D\}$ are automatic relations (for the latter one uses Lemma 2.12). It is also a standard result (cf. [8, 27, 32]) that automatic relations are closed under intersection, projection, and complement. From this it follows that $\mathrm{Sep}_{C,D}(R)$ is a regular language.

Proof of Property B). As $\mathrm{Sep}_{C,D}(R) \subseteq R$, the left-to-right inclusion is immediate. To prove the converse inclusion, consider an input-output pair $(u, v)$ in the right hand-side of the equation, namely, $(u, v)$ is a pair in the relation realised by $T$. Let $\rho$ be the lexicographically least accepting run of $T$ such that $\mathrm{in}(\rho) = u$ and $\mathrm{out}(\rho) = v$. By construction, $\rho \in \mathrm{Sep}_{C,D}(R)$ and hence $(u, v)$ also belongs to the left hand-side of the equation.

Proof of Property C). This holds trivially by the definition of $\mathrm{Sep}_{C,D}(R)$. □

We can now present the missing ingredients of the decomposition result. Proposition 3.2 below shows that, for suitable choices of $C$ and $D$ that depend on the valuedness of $T$, $\mathrm{Cover}_{C,D}(T)$ turns out to be $k$-ambiguous.

This will enable the decomposition result via a classical technique that decomposes any $k$-ambiguous automaton/transducer into a union of $k$ unambiguous ones (see Proposition 3.3 further below).

Proposition 3.2. *Let $T$ be a $k$-valued SST and let $C, D, m$ be as in Lemma 2.14 (note that $m$ depends on $k$). The SST $\mathrm{Cover}_{Cm,Dm^2}(T)$ is $k$-ambiguous.*

Proof. We prove the contrapositive of the statement. Assume that $\mathrm{Cover}_{Cm,Dm^2}(T)$ is not $k$-ambiguous, that is, it admits $k+1$ accepting runs $\rho_0, \ldots, \rho_k$ on the same input. Recall from Proposition 3.1 that for all $0 \leq i < j \leq k$, either $\mathrm{out}(\rho_i) \neq \mathrm{out}(\rho_j)$ or $Cm\text{-delay}(\rho_i, \rho_j) > Dm^2$. By Lemma 2.14 we can find pumped versions of the runs $\rho_0, \ldots, \rho_k$ that have all the same input but have pairwise different outputs, and thus $T$ is not $k$-valued. □

Proposition 3.3. *For all $k \in \mathbb{N}$, every $k$-ambiguous SST can be decomposed into a union of $k$ unambiguous SSTs.*

Proof. The decomposition is done via a classical technique applicable to $k$-ambiguous NFA and, by extension, to all variants of automata and transducers (see [31, 38]). More precisely, decomposing a $k$-ambiguous NFA into a union of $k$ unambiguous NFA is done by ordering runs lexicographically and by letting the $i$-th NFA in the decomposition guess the $i$-th accepting run on a given input (if it exists). Since the lexicographic order is a regular property of pairs of runs, it is easy to track all smaller runs. □

*Proof of Theorem 1.2.* We now have all the ingredients to prove Theorem 1.2, which directly follows from Propositions 3.2 and 3.3, and the fact that unambiguous SSTs can be determinized [5].

As a consequence of Theorem 1.2, we can now prove the correspondence between finite-valued SSTs and finite-valued two-way transducers stated in the introduction as the following theorem:

Theorem 1.3. *Let $R \subseteq \Sigma^* \times \Sigma^*$ be a finite-valued relation. If $R$ can be realized by an SST, then an equivalent two-way transducer can be effectively constructed, and vice-versa.*

Proof. If $R$ is realized by an SST $T$ then we can apply Theorem 1.2 in order to obtain $k$ unambiguous SSTs $T_i$, such that $T$ and the union of $T_1, \ldots, T_k$ are equivalent. From [2] we know that in the functional case, SSTs and two-way transducers are equivalent. Thus, every $T_i$ can be transformed effectively into an equivalent, even deterministic, two-way transducer. From this we obtain an equivalent $k$-ambiguous two-way transducer.

For the converse we start with a $k$-valued two-way transducer $T$ and first observe that we can normalise $T$ in such a way that the crossing sequences[4] of accepting runs of $T$ are bounded by a constant linear in the size of $T$. Once we work with runs with bounded crossing sequences we can construct an equivalent SST in the same way as we do for deterministic two-way transducers. The idea is that during the run of the SST the variables record the outputs generated by the pieces of runs at the left of the current input position (see e.g. [16, 33] for self-contained proofs). □

## 4 FINITE VALUEDNESS

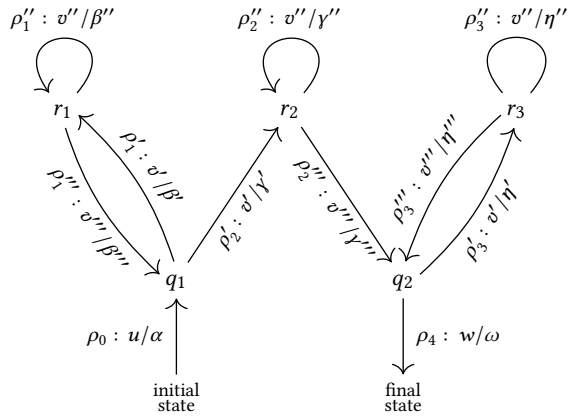
We characterize finite valuedness of SSTs by excluding certain types of substructures. Our characterization has strong analogies with the characterization of finite valuedness for one-way transducers, where the excluded substructures have the shape of a "W" and are therefore called *W-patterns* (cf. [17]).[5]

*Definition 4.1.* A *W-pattern* is a substructure of an SST consisting of states $q_1, q_2, r_1, r_2, r_3$, and some initial and final states, that are

[4] A crossing sequence is a standard notion in the theory of finite-state two-way machines [42], and is defined as the sequence of states in which a given input position is visited by a run of the machine

[5] In [17] there were also other substructures excluded, which however can be seen as degenerate cases of W-patterns.

connected by runs as in the diagram




where a notation like $\rho : u'/\mu$ describes a run named $\rho$ that consumes an input $u'$ and produces an update $\mu$. Moreover, the cyclic runs $\rho_1''$, $\rho_2''$, $\rho_3''$, $\rho_1'\rho_1''\rho_1'''$, $\rho_2'\rho_2''\rho_2'''$, and $\rho_3'\rho_3''\rho_3'''$ are required to be loops, namely, their updates must have idempotent skeletons.

An important constraint of the above definition is that the small loops at states $r_1, r_2, r_3$ consume the same input, i.e. $v''$, and, similarly, the big loops at $q_1$ and $q_2$, as well as the runs from $q_1$ to $q_2$, consume the same set of inputs, i.e. $v'\,(v'')^*\,v'''$.

Given a W-pattern $P$ and a positive natural number $x$, we construct the following runs by composing together copies of the runs of the diagram of Definition 4.1:

$$\begin{aligned} \mathrm{lft}_P^x &= \rho_1'(\rho_1'')^x\,\rho_1''' \\ \mathrm{mid}_P^x &= \rho_2'\,(\rho_2'')^x\,\rho_2''' \\ \mathrm{rgt}_P^x &= \rho_3'\,(\rho_3'')^x\,\rho_3'''. \end{aligned}$$

Similarly, given a sequence $s = (x_1, x_2, \ldots, x_{i-1}, \underline{x_i}, x_{i+1}, \ldots, x_n)$ of positive numbers with exactly one element underlined (we call such a sequence a *marked sequence*), we define the accepting run

$$\mathrm{run}_P(s) = \rho_0 \underbrace{\mathrm{lft}_P^{x_1}\,\mathrm{lft}_P^{x_2}\,\ldots\,\mathrm{lft}_P^{x_{i-1}}}_{\text{loops at } q_1}\,\mathrm{mid}_P^{x_i}\,\underbrace{\mathrm{rgt}_P^{x_{i+1}}\,\mathrm{rgt}_P^{x_{i+2}}\,\ldots\,\mathrm{rgt}_P^{x_n}}_{\text{loops at } q_2}\,\rho_4.$$

For each marked sequence $s = (x_1, \ldots, x_{i-1}, \underline{x_i}, x_{i+1}, \ldots, x_n)$, $\mathrm{run}_P(s)$ consumes the input $u\,v'(v'')^{x_1}v'''\,\ldots\,v'(v'')^{x_n}v'''\,w$ and produces an output of the form

$$\begin{aligned} \mathrm{out}(\mathrm{run}_P(s)) = \big(\iota\,\alpha\,\beta'(\beta'')^{x_1}\beta'''\,\ldots\,\beta'(\beta'')^{x_{i-1}}\beta''' \\ \gamma'(\gamma'')^{x_i}\gamma''' \\ \eta'(\eta'')^{x_{i+1}}\eta'''\,\ldots\,\eta'(\eta'')^{x_n}\eta'''\,\omega\,\omega'\big)(X_1) \end{aligned}$$

where $\iota$ is the initial update and $\omega'$ is the final update determined by the final state of $\mathrm{run}_P(s)$. Note that, differently from the output, the input only depends on the *unmarked sequence*, and thus a W-pattern can have accepting runs that consume the same input and produce arbitrarily many different outputs. As an example, consider a W-pattern as in Definition 4.1, where $\gamma''$ is the only update that produces output symbols – say $\gamma''$ appends letter $c$ to the right of the unique variable. Further suppose that $u = w = \varepsilon$, $v' = v''' = a$, and $v'' = b$. So, on input $(aba)\,(ab^2a)\,\ldots\,(ab^na)$, this W-pattern produces $n$ different outputs: $c, c^2, \ldots, c^n$. The definition and the lemma below generalize this example.

*Definition 4.2.* A W-pattern $P$ is called *divergent* if there exists a 5-tuple of numbers $n_1, n_2, n_3, n_4, n_5 \in \mathbb{N}_+$ for which the two runs $\mathrm{run}_P((n_1, n_2, n_3, \underline{n_4}, n_5))$ and $\mathrm{run}_P((n_1, \underline{n_2}, n_3, n_4, n_5))$ produce different outputs (remark that the runs consume the same input). It is called *simply divergent* if in addition $n_1, n_2, n_3, n_4, n_5 \in \{1, 2\}$.

Theorem 4.3. *An SST is finite-valued iff it does not admit a simply divergent W-pattern.*

The two implications of the theorem are shown in Sections 4.2 and 4.3; effectiveness of the characterization is shown in the next section.

## 4.1 Effectiveness of finite valuedness

We prove that the characterization of finite valuedness in terms of absence of simply divergent W-patterns (Theorem 4.3) is effective, and yields a PSpace decision procedure. We also prove that the equivalence problem for deterministic SSTs, known to be in PSpace, is polynomially reducible to the finite valuedness problem. Despite recent efforts by the community to better understand the complexity of the equivalence problem, it is unknown whether the PSpace upper bound for equivalence (and hence for finite valuedness) can be improved, as no non-trivial lower bound is known. On the other hand, equivalence (as well as finite valuedness) turns out to be in PTime when the number of variables is fixed [5].

The effectiveness procedure uses the following complexity result on the composition of deterministic SSTs, which is of independent interest. It is known that deterministic SSTs are closed under composition because of their equivalence to MSO transductions [2]. A precise complexity analysis of this closure property can be found in [6], but it is triply exponential. We show how to do composition with a single exponential, using results from [15].

Proposition 4.4. *Let $T_1$ and $T_2$ be two deterministic SSTs realizing the functions $f_1 : \Sigma_1^* \to \Sigma_2^*$ and $f_2 : \Sigma_2^* \to \Sigma_3^*$. Let $n_i$ (resp. $m_i$) be the number of states (resp. variables) of $T_i$, and $M = n_1 + n_2 + m_1 + m_2$ One can construct in time exponential in $M$ and polynomial in $|\Sigma_1| + |\Sigma_2| + |\Sigma_3|$ a deterministic SST realizing $f_1 \circ f_2$, with exponentially many states and polynomially many variables in $M$.*

Proof. Each $T_i$ can be converted in polynomial time into an equivalent two-way transducer that is *reversible*, namely, both deterministic and co-deterministic [15]. Again by [15], reversible two-way transducers can be composed in polynomial time, yielding a reversible two-way transducer $S$ realizing $f_1 \circ f_2$, and having polynomially many states in $M$. Finally, it suffices to convert $S$ back to a deterministic SST, which can be done in time exponential in the number of states of $S$ and polynomial in the size of the alphabets. This yields a deterministic SST with exponentially many states and polynomially many variables in $M$ (see e.g. [16, 33]). □

Theorem 1.5. *Given any SST $T$, we can decide in* PSpace if $T$ is finite-valued, and in PTime if the number of variables is fixed. Moreover, this problem is at least as hard as the equivalence problem for deterministic SSTs.

Proof. We start with an overview of the proof. By Theorem 4.3, it suffices to decide whether a given SST $T$ admits a simply divergent W-pattern. Let us fix some tuple $\overline{x} = (x_1, \dots, x_5) \in \{1,2\}^5$. We construct an SST $T_{\overline{x}}$ which is *not* single-valued iff $T$ has a W-pattern which is simply divergent for $\overline{x}$. Since checking single-valuedness of SST is decidable [5], we can decide finite valuedness of $T$ by solving single-valuedness problems for all SST $T_{\overline{x}}$, for all tuples $\overline{x} \in \{1,2\}^5$. Intuitively, we exhibit an encoding of W-patterns $P$ as words $u_P$, and show that the set of such encodings forms a regular language. The encoding $u_P$ informally consists of the runs that form the W-pattern $P$, and some of these runs are overlapped to be able to check that they are on the same input. Accordingly, the SST $T_{\overline{x}}$ will take as input such an encoding $u_P$ and produce as outputs the two words $\mathrm{out}(\mathrm{run}_P(s))$ and $\mathrm{out}(\mathrm{run}_P(s'))$, where $s = (x_1, x_2, x_3, \underline{x_4}, x_5)$ and $s' = (x_1, \underline{x_2}, x_3, x_4, x_5)$. To achieve this, $T_{\overline{x}}$ can consume the input $u_P$ while iterating the encoded runs as prescribed by $s$ or $s'$ and simulating the transitions to construct the appropriate outputs. Finally, an analysis of the size of $T_{\overline{x}}$ and of the algorithm from [5] for checking single-valuedness gives the PSpace upper bound.

*Detailed reduction.* We now explain in detail the reduction to the single-valuedness problem. We will then show how to derive the PSpace upper bound by inspecting the decidability proof for single-valuedness.

Let $T = (\Sigma, \mathcal{X}, Q, Q_{\mathrm{init}}, Q_{\mathrm{final}}, \Omega, \Delta)$ be the given SST and let $\mathcal{P}$ be the set of all W-patterns of $T$. We first show that $\mathcal{P}$ is a regular set, modulo some well-chosen encodings of W-patterns as words. Recall that a W-pattern consists of a tuple of runs $P = (\rho_0, \rho_1', \rho_1'', \rho_1''', \rho_2', \rho_2'', \rho_2''', \rho_3', \rho_3'', \rho_3''', \rho_4)$, connected as in the diagram of Definition 4.1. Note that some of those runs share a common input (e.g. $\rho_1'', \rho_2'', \rho_3''$ share the input $v''$). Therefore, we cannot simply encode $P$ as a the sequence of runs $\rho_0, \rho_1', \dots$, as otherwise regularity would be lost. Instead, in the encoding we overlap groups of runs over the same input, precisely, the group $\{\rho_1', \rho_2', \rho_3'\}$ on input $v'$, the group $\{\rho_1'', \rho_2'', \rho_3''\}$ on input $v''$, and the group $\{\rho_1''', \rho_2''', \rho_3'''\}$ on input $v'''$. Formally, this is done by taking the convolution of the runs in each group, which results in a word over the alphabet $\Delta^3$. Accordingly, $P$ is encoded as the word

$$u_P = \rho_0 \# (\rho_1' \otimes \rho_2' \otimes \rho_3') \# (\rho_1'' \otimes \rho_2'' \otimes \rho_3'') \# (\rho_1''' \otimes \rho_2''' \otimes \rho_3''') \# \rho_4$$

where # is a fresh separator. The language $L_{\mathcal{P}} = \{u_P \mid P \in \mathcal{P}\}$, consisting of all encodings of W-patterns, is easily seen to be regular, recognizable by some automaton $A_{\mathcal{P}}$ which checks that runs forming each convolution share the same input and verify skeleton idempotency (recall that skeletons form a finite monoid). The number of states of the automaton $A_{\mathcal{P}}$ turns out to be polynomial in the number of states of $T$ and in the size of the skeleton monoid, which in turn is exponential in the number of variables.

Next, we construct the SST $T_{\overline{x}}$ as the disjoint union of two deterministic SSTs $T_s$ and $T_{s'}$, where $s = (x_1, x_2, x_3, \underline{x_4}, x_5)$ and $s' = (x_1, \underline{x_2}, x_3, x_4, x_5)$. We only describe $T_s$, as the construction of $T_{s'}$ is similar. The SST $T_s$ is obtained as a suitable restriction of the composition of two deterministic SSTs $T^s_{\mathrm{iter}}$ and $T_{\mathrm{exec}}$, which respectively iterate the runs as prescribed by $s$ and execute the transitions read as input. When fed with the encoding $u_P$ of a W-pattern, $T^s_{\mathrm{iter}}$ needs to output $\mathrm{run}_P(s) \in \Delta^*$. More precisely, it takes as input a word of the form $\rho_0 \# (\rho_1' \otimes \rho_2' \otimes \rho_3') \# (\rho_1'' \otimes \rho_2'' \otimes \rho_3'') \# (\rho_1''' \otimes \rho_2''' \otimes \rho_3''') \# \rho_4$ and produces as output

$$\rho_0\ \rho_1'\ (\rho_1'')^{x_1}\ \rho_1'''\ \rho_1'\ (\rho_1'')^{x_2}\ \rho_1'''\ \rho_1'\ (\rho_1'')^{x_3}\ \rho_1'''$$
$$\rho_2'\ (\rho_2'')^{x_4}\ \rho_2'''$$
$$\rho_3'\ (\rho_3'')^{x_5}\ \rho_3'''\ \rho_4.$$

The SST $T^s_{\mathrm{iter}}$ uses one variable for each non-iterated run (e.g. for $\rho_0$ and $\rho_1'$), $x_1 + x_2 + x_3$ variables to store copies of $\rho_1''$, $x_4$ variables to store copies of $\rho_2''$, and $x_5$ variables to store copies of $\rho_3''$, and eventually outputs the concatenation of all these variables to obtain $\mathrm{run}_P(s)$. Note that $T^s_{\mathrm{iter}}$ does not need to check that the input is a well-formed encoding (this is done later when constructing $T_s$), so the number of its states and variables is bounded by a constant; on the other hand, the input alphabet, consisting of transitions of $T$, is polynomial in the size of $T$.

The construction of $T_{\mathrm{exec}}$ is straightforward: it just executes the transitions it reads along the input, thus simulating a run of $T$. Hence $T_{\mathrm{exec}}$ has a single state and the same number of variables as $T$. Its alphabet is linear in the size of $T$.

Now, $T_s$ is obtained from the composition $T_{\mathrm{exec}} \circ T^s_{\mathrm{iter}}$ by restricting the input domain to $\mathcal{P}$. It is well-known that deterministic SST are closed under composition and regular domain restriction [2]. By the above constructions, we have

$$T_{\bar{x}}(u_P) = T_s(u_P) \cup T_{s'}(u_P) = \{\mathrm{out}(\mathrm{run}_P(s)) \cup \{\mathrm{out}(\mathrm{run}_P(s'))\}$$

and hence $T$ contains a W-pattern that is simply divergent for $\bar{x}$ iff $T_{\bar{x}}$ is not single-valued. This already implies the decidability of the existence of a simply divergent W-pattern in $T$, and hence by Theorem 4.3, of finite valuedness.

*Complexity analysis.* Let us now analyse the complexity in detail. This requires first estimating the size of $T_{\bar{x}}$. Let $n_T$ resp. $m_T$ be the number of states of $T$, resp. its number of variables. From the previous bounds on the sizes of $T_{\mathrm{exec}}$ and $T^s_{\mathrm{iter}}$ and Proposition 4.4, we derive that the number of states and variables of $T_{\mathrm{exec}} \circ T^s_{\mathrm{iter}}$ is polynomial in both $n_T$ and $m_T$. Further, restricting the domain to $\mathcal{P}$ is done via a product with the automaton $A_{\mathcal{P}}$, whose size is polynomial in $n_T$ and exponential in $m_T$. Summing up, the number of states of $T_s$ is exponential in $m_T$, and polynomial in $n_T$. Its number of variables is polynomial in both $n_T$ and $m_T$. And so do $T_{s'}$ and $T_{\bar{x}}$.

As explained in [5], checking single-valuedness of SST reduces to checking non-emptiness of a 1-reversal 2-counter machine of size exponential in the number of variables and polynomial in the number of states. This is fortunate, since it allows us to conclude that checking single-valuedness of $T_{\bar{x}}$ reduces to checking non-emptiness of a 1-reversal 2-counter machine of size just exponential in the number of variables of $T$. The PSpace upper bound (and the PTime upper bound for a fixed number of variables) now follow by recalling that non-emptiness of counter machines with fixed numbers of reversals and counters is in NLogSpace [25].

*Lower bound.* For the lower bound, consider two deterministic SST $T_1, T_2$ over some alphabet $\Sigma$ with same domain $D$. Domain equivalence can be tested in PTime because $T_1, T_2$ are deterministic. Consider a fresh symbol $\# \notin \Sigma$, and the relation

$$R = \left\{ \left(u_1 \# \dots \# u_n,\ T_{i_1}(u_1) \# \dots \# T_{i_n}(u_n)\right) \;\middle|\; \begin{array}{c} u_i \in D,\ n \in \mathbb{N}, \\ i_1, \dots, i_n \in \{1,2\} \end{array} \right\}$$

It is easily seen that $R$ is realizable by a (non-deterministic) SST. We claim that $R$ is finite-valued iff it is single-valued, iff $T_1$ and $T_2$ are equivalent. If $T_1$ and $T_2$ are equivalent, then $T_1(u_j) = T_2(u_j)$ for all $1 \leq j \leq n$, hence $R$ is single-valued, and so finite-valued. Conversely, if $T_1$ and $T_2$ are not equivalent, then $T_1(u) \neq T_2(u)$ for some $u \in D$, and the family of inputs $(u\#)^n\, u$, with $n \in \mathbb{N}$, witnesses the fact that $R$ is not finite-valued. □

As a corollary, we obtain an alternative proof of the following known result that was recalled in the introduction:

Corollary 1.6 ([46]). *Finite-valuedness of two-way transducers is decidable in* PSpace.

Proof. Observe that a necessary condition for a two-way transducer to be finite-valued is that crossing sequences are bounded. More precisely, if a crossing sequence has a loop then the output of the loop must be empty, otherwise the transducer is not finite valued. Given a bound on the length of crossing sequences the standard conversion into an equivalent SST applies, see e.g. [16, 33]. This yields an SST with an exponential number of states and a linear number of variables, both in the number of states of the initial two-way transducer. Finally, we apply the algorithm of Theorem 1.5, and we observe that it amounts to checking emptiness of a 1-reversal 2-counter machine whose number of states is exponential in the number of states of the initial two-way transducer. We again conclude by using the NLogSpace algorithm for checking emptiness of such counter machines [25]. □

## 4.2 A necessary condition for finite valuedness

Here we prove the contrapositive of the left-to-right implication of Theorem 4.3: we show that in a divergent W-pattern there exist arbitrarily many outputs produced by the same input.

Lemma 4.5. *Every SST that contains some divergent W-pattern is not finite-valued.*

Proof. Let us fix an SST with a divergent W-pattern $P$. In order to prove that the SST is not finite-valued, we show that we can construct arbitrary numbers of accepting runs of $P$ that consume the same input and produce pairwise different outputs. To do this we will consider for some suitable $M \in \mathbb{N}$ inequalities in the formal parameters $\mathsf{s}_1, \ldots, \mathsf{s}_M$ (where $1 \leq i < j \leq M$), and look for arbitrary large, satisfiable, subsets of such inequalities:

$$e_{M,i,j}[\mathsf{s}_1, \mathsf{s}_2, \ldots, \mathsf{s}_M]: \quad \mathrm{out}(\mathrm{run}_P(\mathsf{s}_1, \mathsf{s}_2, \ldots, \mathsf{s}_{i-1}, \underline{\mathsf{s}_i}, \mathsf{s}_{i+1}, \ldots, \mathsf{s}_M)) \neq \mathrm{out}(\mathrm{run}_P(\mathsf{s}_1, \mathsf{s}_2, \ldots, \mathsf{s}_{j-1}, \underline{\mathsf{s}_j}, \mathsf{s}_{j+1}, \ldots, \mathsf{s}_M)).$$

Recall that, according to the diagram of Definition 4.1, the number of variable occurrences before (resp. after) the underlined parameter represents the number of loops at state $q_1$ (resp. $q_2$) in a run of the W-pattern. Moreover, each variable $\mathsf{s}_i$ before (resp. after) the underlined parameter represents the number of repetitions of the loop at $r_1$ (resp. $r_3$) within occurrences of bigger loops at $q_1$ (resp. $q_2$); similarly, the underlined variable represents the number of repetitions of the loop at $r_2$ within the run that connects $q_1$ to $q_2$. In view of this, by Corollary 2.6, the outputs of the runs considered in the above inequality have the format required for a word inequality with repetitions parametrized by $\mathsf{s}_1, \ldots, \mathsf{s}_M$.

The fact that the W-pattern $P$ is divergent allows to find sets of satisfiable inequalities $e_{M,i,j}$ of arbitrary large cardinality. This in turn will yield together with our word combinatorics results, arbitrary numbers of accepting runs over the same input and with pairwise different outputs.

**Claim.** *For every $m \in \mathbb{N}$, there exist $M \in \mathbb{N}$ and a set $I \subseteq \{1, 2, \ldots, M\}$ of cardinality $m+1$ such that, for all $i < j \in I$, $e_{M,i,j}$ is satisfiable.*

Proof of the claim. Since $P$ is a divergent W-pattern, there exist $n_1, n_2, n_3, n_4, n_5 \in \mathbb{N}_+$ such that

$$\mathrm{out}(\mathrm{run}_P(n_1, \underline{n_2}, n_3, n_4, n_5)) \neq \mathrm{out}(\mathrm{run}_P(n_1, n_2, n_3, \underline{n_4}, n_5)).$$

We fix such numbers $n_1, n_2, n_3, n_4, n_5 \in \mathbb{N}_+$. Consider now the following inequality over the formal parameters $\mathsf{x}, \mathsf{y}, \mathsf{z}$:

$$e[\mathsf{x}, \mathsf{y}, \mathsf{z}]: \quad \mathrm{out}(\mathrm{run}_P(\overbrace{n_1, n_1, \ldots, n_1}^{\text{x times}}, \underline{n_2}, \overbrace{n_3, \ldots, n_3}^{\text{y times}}, n_4, \overbrace{n_5, \ldots, n_5}^{\text{z times}})) \neq \mathrm{out}(\mathrm{run}_P(\underbrace{n_1, n_1, \ldots, n_1}_{\text{x times}}, n_2, \underbrace{n_3, \ldots, n_3}_{\text{y times}}, \underline{n_4}, \underbrace{n_5, \ldots, n_5}_{\text{z times}})).$$

Note that every instance of $e[\mathsf{x}, \mathsf{y}, \mathsf{z}]$ with concrete values $x, y, z$ is also an instance of $e_{M,i,j}$, where $M = x + y + z + 2$, $i = x + 1$, $j = x + y + 2$, and all parameters $\mathsf{s}_1, \ldots, \mathsf{s}_M$ are instantiated with values from $\{n_1, \ldots, n_5\}$. Moreover, as the parameters in $e[\mathsf{x}, \mathsf{y}, \mathsf{z}]$ determine the number of repetitions of $n_1, n_3, n_5$, which in their turn correspond to pumping loops at $q_1$ and $q_2$, by Corollary 2.6, the outputs of the considered runs have the format required for a word inequality with repetitions parametrized by $\mathsf{x}, \mathsf{y}, \mathsf{z}$.

Since $e[\mathsf{x}, \mathsf{y}, \mathsf{z}]$ is satisfiable (e.g. with $\mathsf{x} = \mathsf{y} = \mathsf{z} = 1$), Corollary 2.10 implies that

$$\exists \ell_{\mathsf{y}}\, \forall h_{\mathsf{y}}\, \exists \ell_{\mathsf{x}}\, \forall h_{\mathsf{x}}\, \exists \ell_{\mathsf{z}}\, \forall h_{\mathsf{z}} \quad \underbrace{[\ell_{\mathsf{x}}, h_{\mathsf{x}}]}_{\text{values for x}} \times \underbrace{[\ell_{\mathsf{y}}, h_{\mathsf{y}}]}_{\text{values for y}} \times \underbrace{[\ell_{\mathsf{z}}, h_{\mathsf{z}}]}_{\text{values for z}} \subseteq \mathrm{Sols}(e).$$

Note that we start by quantifying over $\ell_y$ and not $\ell_x$ (Corollary 2.10 is invariant with respect to the parameter order)[6]. This particular order ensures that for every $m \in \mathbb{N}$ there exist three integers $\ell_y, \ell_x, \ell_z > 0$ such that

$$[\ell_{\mathsf{x}}, \ell_{\mathsf{x}} + 2m\ell_{\mathsf{y}}] \times [\ell_{\mathsf{y}}, 2m\ell_{\mathsf{y}}] \times [\ell_{\mathsf{z}}, \ell_{\mathsf{z}} + 2m\ell_{\mathsf{y}}] \subseteq \mathrm{Sols}(e). \quad (2)$$

Further note that $h_y = 2m\ell_y$ depends only on $\ell_y$, while $h_x = \ell_x + 2m\ell_y$ depends on both $\ell_x$ and $\ell_y$.

We can now prove the claim by letting $M = \ell_{\mathsf{x}} + 2m\ell_{\mathsf{y}} + \ell_{\mathsf{z}} + 1$ and $I = \{\ell_{\mathsf{x}} + 2\lambda\ell_{\mathsf{y}} + 1 \mid 0 \leq \lambda \leq m\}$. Indeed, the gap between two consecutive values of $I$ equals $2\ell_y$, and for every $i < j \in I$ we get

$$\begin{array}{lcll} i-1 & \in & [\ell_{\mathsf{x}}, \ell_{\mathsf{x}} + 2(m-1)\ell_{\mathsf{y}}] & \subseteq [\ell_{\mathsf{x}}, \ell_{\mathsf{x}} + 2m\ell_{\mathsf{y}}] \\ j-i-1 & \in & [2\ell_{\mathsf{y}} - 1, 2m\ell_{\mathsf{y}} - 1] & \subseteq [\ell_{\mathsf{y}}, 2m\ell_{\mathsf{y}}] \\ M-j & \in & [\ell_{\mathsf{z}}, \ell_{\mathsf{z}} + 2(m-1)\ell_{\mathsf{y}}] & \subseteq [\ell_{\mathsf{z}}, \ell_{\mathsf{z}} + 2m\ell_{\mathsf{y}}]. \end{array}$$

[6] The reader may compare this with the example given before Def. 4.2.

Thus, by Equation (2), $(i-1, j-i-1, M-j) \in \mathrm{Sols}(e)$. This solution of $e$ corresponds to the instance of $e_{M,i,j}$ with the values for the formal parameters $\mathsf{s}_1, \ldots, \mathsf{s}_M$ defined by

$$\mathsf{s}_h = \begin{cases} n_1 & \text{for every } 1 \le h \le i-1, \\ n_2 & \text{for } h = i, \\ n_3 & \text{for every } i+1 \le h \le j-1, \\ n_4 & \text{for } h = j, \\ n_5 & \text{for every } j+1 \le M. \end{cases}$$

Hence, $e_{M,i,j}$ is satisfiable for all $i < j \in I$, as claimed. □

We can now conclude the proof of the lemma using the above claim: Corollary 2.9 tells us that any system of word inequalities is satisfiable when every word inequality in it is so. Using this and the above claim, we derive that for every $m$ there exist $t_1, t_2, \ldots, t_M \in \mathbb{N}_+$ such that, for all $i < j \in I$ (with $I$ as in the claim), $e_{M,i,j}[t_1, t_2, \ldots, t_M]$ holds. For every $h \in I$, let

$$\rho_h \;=\; \mathrm{run}_P(t_1, t_2, \ldots, t_{h-1}, \underline{t_h}, t_{h+1}, \ldots, t_M).$$

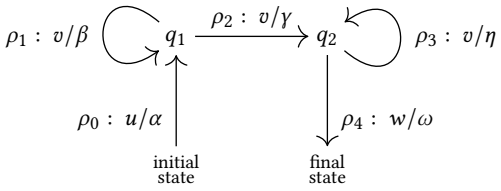
Note that all runs $\rho_h$, for $h \in I$, consume the same input, since they all correspond to the same unmarked sequence $(t_1, t_2, \ldots, t_M)$. However, they produce pairwise different outputs, because for every $i < j \in I$, the tuple $(t_1, t_2, \ldots, t_M)$ is a solution of $e_{M,i,j}$. Since $|I| = m$ can be chosen arbitrarily, the transducer is not finitely valued. □

## 4.3 A sufficient condition for finite valuedness

We finally prove that an SST with no simply divergent W-pattern is finite-valued. The proof relies on two crucial results.

The first result is a characterizastion of finite ambiguity for SSTs, which is easily derived from a prior characterization of finite ambiguity for finite state automata [29, 34, 45]:

*Definition 4.6.* A *dumbbell* is a substructure of an SST consisting of states $q_1, q_2$ connected by runs as in the diagram



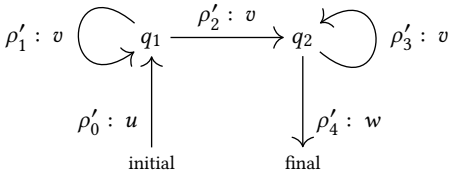

where the runs $\rho_1$ and $\rho_3$ are loops (in particular, they produce updates with idempotent skeletons) and at least two among the runs $\rho_1, \rho_2, \rho_3$ that consume the same input $v$ are distinct.

Lemma 4.7. *An SST is finitely ambiguous iff it does not contain any dumbbell.*

Proof. Let $T = (\Sigma, \mathcal{X}, Q, Q_{\mathrm{init}}, Q_{\mathrm{final}}, \Omega, \Delta)$ be an SST. By projecting away the updates on the transitions we obtain from $T$ a *multiset finite state automaton* $A$. Formally, $A = (\Sigma, Q_{\mathrm{init}}, Q_{\mathrm{final}}, \Delta')$, where $\Delta'$ is the *multiset* containing one occurrence of a triple $(q, a, q')$ for each transition of the form $(q, a, \alpha, q')$ in $\Delta$. Note that a multiset automaton can admit several occurrences of the same (accepting) run. Accordingly, the notion of finite ambiguity for $A$ requires the existence of a uniform bound to the number of *occurrences* of accepting runs of $A$ on the same input. We also remark that multiset automata are essentially the same as weighted automata over the semiring of natural numbers (the weight of a transition being its number of occurrences), with only a difference in terminology where finite ambiguity in multiset automata corresponds to finite valuedness in weighted automata.

Given the above construction of $A$ from $T$, one can verify by induction on $|u|$ that the number of occurrences of accepting runs of $A$ on $u$ coincides with the number of accepting runs of $T$ on $u$. This means that $A$ is finitely ambiguous iff $T$ is finitely ambiguous.

Finally, we recall the characterizations of finite ambiguity from [29, 34, 45] (see in particular Theorem 1.1 and Lemma 2.6 from [34]). For short, their results directly imply that a multiset automaton is finitely ambiguous iff it does not contain a *plain dumbbell*, namely, a substructure of the form




where at least two among $\rho'_1, \rho'_2, \rho'_3$ are distinct runs.

This almost concludes the proof of the lemma, since any dumbbell of $T$ can be projected into a plain dumbbell of $A$. The converse implication, however, is not completely straightforward. The reason is that the cyclic runs of a plain dumbbell in $A$ do not necessarily correspond to loops in the SST $T$, as the runs need not produce updates with idempotent skeletons. Nonetheless, we can reason as follows. Suppose that $A$ contains a plain dumbbell, with occurrences of runs $\rho'_0, \rho'_1, \rho'_2, \rho'_3, \rho'_4$ as depicted above. Let $\rho_0, \rho_1, \rho_2, \rho_3, \rho_4$ be the corresponding runs in $T$, and let $\alpha, \beta, \gamma, \eta, \omega$ be their induced updates. Further let $n$ be a large enough number such that $\beta^n$ and $\eta^n$ have idempotent skeletons (such an $n$ always exists since the skeleton monoid is finite). Now consider the substructure in $T$ given by the runs $\rho_0 = \rho'_0$, $\rho_1 = (\rho'_1)^n$, $\rho_2 = (\rho'_1)^{n-1}\,\rho'_2$, $\rho_3 = (\rho'_3)^n$, and $\rho_4 = \rho'_4$. This substructure satisfies precisely the definition of dumbbell for the SST $T$. □

The second ingredient for the proof of the right-to-left implication of Theorem 4.3 uses once more the cover construction described in Proposition 3.1. More precisely, in Lemma 4.8 below we show that if an SST $T$ has no simply divergent W-pattern, then, for some well chosen values $C$, $D$, $m$, the SST $\mathrm{Cover}_{Cm,Dm^2}(T)$ contains no dumbbell. Before proving the lemma, let us show how it can be used to establish the right-to-left implication of Theorem 4.3.

*Proof of Theorem 4.3.* By Lemma 4.7, if $\mathrm{Cover}_{Cm,Dm^2}(T)$ has no dumbbell, then it is finitely ambiguous, and hence it associates with each input a uniformly bounded number of outputs. In particular, $\mathrm{Cover}_{Cm,Dm^2}(T)$ is finite-valued, and since it is also equivalent to $T$, then $T$ is finite-valued too. □

Lemma 4.8. *Given an SST $T$, one can compute three numbers $C$, $D$, and $m$ such that if $\mathrm{Cover}_{Cm,Dm^2}(T)$ contains a dumbbell, then $T$ contains a simply divergent W-pattern.*

Proof. Intuitively, we will show that if $\mathrm{Cover}_{Cm,Dm^2}(T)$ contains a dumbbell, then this dumbbell must admit two distinct runs $\pi, \pi'$ that either produce different outputs or have large delay. In

both cases we will be able to transform the dumbbell into a simply divergent W-pattern. For example, in the case of a large delay, we will rely on Lemmas 2.13 and 2.4 to identify certain loops in $\pi$ and $\pi'$ that lie entirely inside occurrences of the runs $\rho_0, \dots, \rho_4$ of the dumbbell, and such that, when pumped, produce different outputs. From there, we will be able to expose a simply divergent W-pattern in $\mathrm{Cover}_{Cm,Dm^2}(T)$, and hence in $T$ as well.

Formally, let $T$ be an SST, let $C, D$ be defined as in Lemma 2.13, and let $m = 7E^{H^2+H+1} + 1$, where $E, H$ are defined as in Lemma 2.4. Next, suppose that $\mathrm{Cover}_{Cm,Dm^2}(T)$ contains a dumbbell like the one of Definition 4.6, with runs $\rho_0, \rho_1, \rho_2, \rho_3, \rho_4$ that produce respectively the updates $\alpha, \beta, \gamma, \eta, \omega$.

Consider the following accepting runs, which are obtained by composing the copies of the original runs of the dumbbell, and that are different because two of $\rho_1, \rho_2, \rho_3$ have to be different:

$$\begin{aligned} \pi &= \rho_0\ \rho_1\ \rho_2\ \rho_3\ \rho_3\ \rho_3\ \rho_4 \\ \pi' &= \rho_0\ \rho_1\ \rho_1\ \rho_1\ \rho_2\ \rho_3\ \rho_4. \end{aligned} \tag{3}$$

By the properties of $\mathrm{Cover}_{Cm,Dm^2}(T)$, since $\pi$ and $\pi'$ consume the same input, they either produce different outputs or have $Cm$-delay larger than $Dm^2$.

We first consider the case where the outputs are different. In this case, we can immediately witness a simply divergent W-pattern $P$ by adding empty runs to the dumbbell; formally, for every $i = 1, 2, 3$, we let $\rho'_i = \rho_i$ and $\rho''_i = \rho'''_i = \varepsilon$, so as to form a W-pattern like the one in the diagram of Definition 4.1, but now with $r_1 = q_1$ and $r_2 = r_3 = q_2$. Using the notation introduced at the beginning of Section 4, we observe that $\pi = \mathrm{run}_P(1, \underline{1}, 1, 1, 1)$ and $\pi' = \mathrm{run}_P(1, 1, 1, \underline{1}, 1)$ — recall that the underlined number represents how many times the small loop at $r_2$, which is empty here, is repeated along the run from $q_1$ to $q_2$, and the other numbers represent how many times the small loops at $r_1$ and $r_3$, which are also empty here, are repeated within the occurrences of big loops at $q_1$ and $q_2$. Since, by assumption, the runs $\pi$ and $\pi'$ produce different outputs, the W-pattern $P$ is simply divergent, as required.

We now consider the case where $\pi$ and $\pi'$ have large delay, namely, $Cm\text{-delay}(\pi, \pi') > Dm^2$. In this case Lemma 2.13 guarantees the existence of a set $I \subseteq \{0, 1, \dots, |\pi|\}$ containing $m$ positions in between the input letters such that, for all pairs $i < j$ in $I$, the interval $[i, j]$ is a loop on both $\pi$ and $\pi'$ and satisfies

$$\mathrm{out}(\mathrm{pump}^2_{[i,j]}(\pi)) \neq \mathrm{out}(\mathrm{pump}^2_{[i,j]}(\pi')). \tag{4}$$

Next, recall from Equation (3) that $\pi$, and similarly $\pi'$, consists of seven parts, representing copies of the original runs of the dumbbell and consuming the inputs $u, v, v, v, v, v, w$. We identify these parts with the numbers $1, \dots, 7$. Since we defined $m$ as $7E^{H^2+H+1} + 1$, there is one of these parts in which at least $E^{H^2+H+1} + 1$ of the aforementioned positions of $I$ occur. Let $p \in \{1, 2, \dots, 7\}$ denote the number of this part, and let $I_p$ be a set of $E^{H^2+H+1}+1$ positions from $I$ that occur entirely inside the $p$-th part. We conclude the proof by a further case distinction, depending on whether $p \in \{1, 7\}$ or $p \in \{2, \dots, 6\}$.

*Parts* 1 *and* 7. Let us suppose that $p \in \{1, 7\}$, and let $i$ and $j$ be two distinct positions in $I_p$. We let $P$ be the W-pattern obtained by transforming the dumbbell as follows:

(1) First, we pump either $\rho_0$ or $\rho_4$ depending on $p$:
   - If $p = 1$, we set $\rho'_0 = \mathrm{pump}^2_{[i,j]}(\rho_0)$ and $\rho'_4 = \rho_4$.
   - If $p = 7$, we set $\rho'_4 = \mathrm{pump}^2_{[i-|u\,v^5|, j-|u\,v^5|]}(\rho_4)$ and $\rho'_0 = \rho_0$.
(2) Then, we add empty runs to $\rho_1, \rho_2$, and $\rho_3$; formally, for each $h \in \{1, 2, 3\}$, we set $\rho'_h = \rho_h$ and $\rho''_h = \rho'''_h = \varepsilon$.

Now that we identified a W-pattern $P$ in $\mathrm{Cover}_{C,D}(T)$, we note that

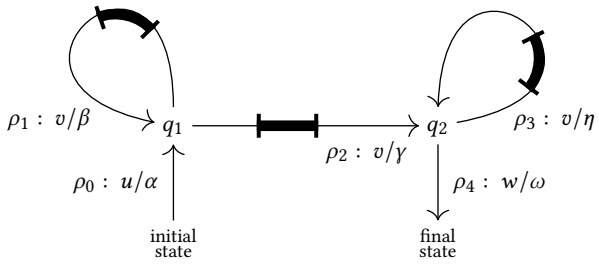
$$\begin{aligned} \mathrm{pump}^2_{[i,j]}(\pi) &= \mathrm{run}_P(1, \underline{1}, 1, 1, 1) \\ \mathrm{pump}^2_{[i,j]}(\pi') &= \mathrm{run}_P(1, 1, 1, \underline{1}, 1). \end{aligned}$$

We also recall Equation 4, which states that these runs produce different outputs. This means that the W-pattern $P$ is simply divergent. Finally, since the runs of $\mathrm{Cover}_{C,D}(T)$ can be projected into runs of $T$, we conclude that $T$ contains a simply divergent W-pattern.

*Parts* $2 - 6$. Let us suppose that $p \in \{2, \dots, 6\}$. Note that, in this case, the elements of $I_p$ denote positions inside the $p$-th factor of the input $u\,v\,v\,v\,v\,v\,w$, which is a $v$. To refer directly to the positions of $v$, we define $I'_p$ as the set obtained by subtracting $|u\,v^{p-1}|$ from each element of $I_p$. Since the set $|I'_p|$ has cardinality $E^{H^2+H+1} + 1$, we claim that we can find an interval with endpoints from $I'_p$ that is a loop of $\rho_1$, $\rho_2$, and $\rho_3$, at the same time. Specifically, we can do so via three consecutive applications of Lemma 2.4:

(1) As $|I'_p| = E^{H^2+H+1} + 1 = E \cdot (E^{H+1})^H + 1$, there exists a set $I''_p \subseteq I'_p$ of cardinality $E^{H+1} + 1$ such that for every pair $i < j$ in $I''_p$, the interval $[i, j]$ is a loop of $\rho_1$;
(2) As $|I''_p| = E^{H+1} + 1 = E \cdot E^H + 1$, there exists $I'''_p \subseteq I''_p$ of cardinality $E + 1$ s.t. for every pair $i < j$ in $I'''_p$, the interval $[i, j]$ is a loop of $\rho_2$ (and also of $\rho_1$, since $i, j \in I'''_p \subseteq I''_p$);
(3) As $|I'''_p| = E + 1 = E \cdot 1^H + 1$, there are two positions $i < j$ in $I'''_p$ such that the interval $[i, j]$ is a loop of $\rho_3$ (and also of $\rho_1$ and $\rho_2$ since $i, j \in I'''_p \subseteq I''_p$).

The diagram below summarizes the current situation: we have just managed to find an interval $[i, j]$ that is a loop on all $v$-labelled runs $\rho_1, \rho_2, \rho_3$ of the dumbbell (the occurrences of this interval inside $\rho_1, \rho_2, \rho_3$ are highlighted by thick segments):




We can now expose a W-pattern $P$ by merging the positions $i$ and $j$ inside each $v$-labelled run $\rho_1, \rho_2$, and $\rho_3$ of the dumbbell. Formally, we let $\rho'_0 = \rho_0$, $\rho'_4 = \rho_4$, and for every $h \in \{1, 2, 3\}$, we define $\rho'_h$, $\rho''_h$, and $\rho'''_h$, respectively, as the intervals $[0, i]$, $[i, j]$, and $[j, |v|]$ of $\rho_h$. Now that we have identified a W-pattern $P$ inside $\mathrm{Cover}_{C,D}(T)$, we remark that $\pi = \mathrm{run}_P(1, \underline{1}, 1, 1, 1)$ and $\pi' = \mathrm{run}_P(1, 1, 1, \underline{1}, 1)$. Additionally, if we transpose $i$ and $j$ from $I'_p$ back to $I$, that is, if we set $i' = i + |u\,v^{p-1}|$ and $j' = j + |u\,v^{p-1}|$, since both $i'$ and $j'$

occur in the $p$-th part of $\pi$ and $\pi'$, pumping the interval $[i', j']$ in $\pi$ (resp. $\pi'$) amounts to incrementing the $(p-1)$-th parameter in the notation $\mathrm{run}_P(1, \underline{1}, 1, 1, 1)$ (resp. $\mathrm{run}_P(1, 1, 1, \underline{1}, 1)$). More precisely:

$$\mathrm{pump}^2_{[i',j']}(\pi) = \mathrm{run}_P(n_1, \underline{n_2}, n_3, n_4, n_5)$$
$$\mathrm{pump}^2_{[i',j']}(\pi') = \mathrm{run}_P(n_1, n_2, n_3, \underline{n_4}, n_5)$$

where each $n_{p'}$ is either 2 or 1 depending on whether $p' = p - 1$ or not. Since Equation 4 states that these two runs produce different outputs, the W-pattern $P$ is simply divergent. Finally, since the runs of $\mathrm{Cover}_{C,D}(T)$ can be projected into runs of $T$, we conclude that, also in this case, $T$ contains a simply divergent W-pattern. □

## 5 CONCLUSION

We have drawn a rather complete picture of finite-valued SSTs and answered several open questions of [5]. Regarding expressiveness, they can be decomposed as unions of deterministic SST (Theorem 1.2), and they are equivalent to finite-valued two-way transducers (Theorem 1.3), and to finite-valued non-deterministic MSO transductions (see Section 1). On the algorithmic side, their equivalence problem is decidable in elementary time (Theorem 1.4) and finite valuedness of SSTs is decidable in PSpace (PTime for a fixed number of variables), see Theorem 1.5. As an alternative proof to the result of [46], our results imply the PSpace decidability of finite valuedness for two-way transducers (Corollary 1.6). Because of the effective expressiveness equivalence between SSTs and non-deterministic MSO transductions, our result also entails decidability of finite valuedness for the latter class.

**Further questions.** A first interesting question is how big the valuedness of an SST can be. In the classical case of one-way transducers the valuedness has been shown to be at most exponential (if finite at all) [43]. We can obtain a bound from Lemma 4.8, but the value is likely to be sub-optimal.

Our equivalence procedure relies on the decomposition of a $k$-valued SST into a union of $k$ deterministic SSTs each of elementary size. Our construction is likely to be sub-optimal again, and so is our complexity for checking equivalence. On the other hand, only a PSpace lower bound is known [35]. A better understanding of the complexity of the equivalence problem for (sub)classes of SSTs is a challenging question. Already for deterministic SSTs, equivalence is only known to be between NLogSpace and PSpace [3].

However, beyond the finite-valued setting there is little hope to find a natural restriction on valuedness which would preserve the decidability of the equivalence problem. Already for one-way transducers of linear valuedness (i.e. where the number of outputs is linear in the input length), equivalence is undecidable, as shown through a small modification of the proof of [28].

Deterministic SSTs have been extended, while preserving decidability of the equivalence problem, in several ways: to copyful SSTs [9, 21], which allow to copy the content of variables several times, to infinite strings [7], and to trees [4]. Generalizations of these results to the finite-valued setting yield interesting questions. On trees, similar questions (effective finite valuedness, decomposition and equivalence) have been answered positively for bottom-up tree transducers [41].

Finally, SSTs have linear input-to-output growth (in the length of the strings). There is a recent trend in extending transducer models to allow polynomial growth [10–12, 18], and finite valuedness has not yet been studied in this context.

## REFERENCES


[1] M.H. Albert and J. Lawrence. 1985. A proof of Ehrenfeucht's conjecture. *Theor. Comput. Sci.* 41, 1 (1985), 121–123.

[2] Rajeev Alur and Pavol Cerný. 2010. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India (LIPIcs, Vol. 8)*, Kamal Lodaya and Meena Mahajan (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 1–12. https://doi.org/10.4230/LIPICS.FSTTCS.2010.1

[3] Rajeev Alur and Pavol Cerný. 2011. Streaming transducers for algorithmic verification of single-pass list-processing programs. In *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, Thomas Ball and Mooly Sagiv (Eds.). ACM, 599–610. https://doi.org/10.1145/1926385.1926454

[4] Rajeev Alur and Loris D'Antoni. 2017. Streaming Tree Transducers. *J. ACM* 64, 5 (2017), 31:1–31:55. https://doi.org/10.1145/3092842

[5] Rajeev Alur and Jyotirmoy V. Deshmukh. 2011. Nondeterministic Streaming String Transducers. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 6756)*, Luca Aceto, Monika Henzinger, and Jirí Sgall (Eds.). Springer, 1–20. https://doi.org/10.1007/978-3-642-22012-8_1

[6] Rajeev Alur, Taylor Dohmen, and Ashutosh Trivedi. 2022. Composing Copyless Streaming String Transducers. *CoRR* abs/2209.05448 (2022), 1–21. https://doi.org/10.48550/ARXIV.2209.05448 arXiv:2209.05448

[7] Rajeev Alur, Emmanuel Filiot, and Ashutosh Trivedi. 2012. Regular Transformations of Infinite Strings. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*. IEEE Computer Society, 65–74. https://doi.org/10.1109/LICS.2012.18

[8] Achim Blumensath and Erich Grädel. 2000. Automatic Structures. In *Proceedings of the 15th IEEE Symposium on Logic in Computer Science, LICS 2000*. IEEE Computer Society Press, 51–62.

[9] Mikolaj Bojanczyk. 2019. The Hilbert method for transducer equivalence. *ACM SIGLOG News* 6, 1 (2019), 5–17. https://doi.org/10.1145/3313909.3313911

[10] Mikolaj Bojanczyk. 2022. Transducers of polynomial growth. In *LICS '22: 37th Annual ACM/IEEE Symposium on Logic in Computer Science, Haifa, Israel, August 2 - 5, 2022*, Christel Baier and Dana Fisman (Eds.). ACM, 1:1–1:27. https://doi.org/10.1145/3531130.3533326

[11] Mikolaj Bojanczyk, Sandra Kiefer, and Nathan Lhote. 2019. String-to-String Interpretations With Polynomial-Size Output. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 106:1–106:14. https://doi.org/10.4230/LIPICS.ICALP.2019.106

[12] Mikołaj Bojańczyk. 2023. On the Growth Rates of Polyregular Functions. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. 1–13. https://doi.org/10.1109/LICS56636.2023.10175808

[13] Bruno Courcelle and Joost Engelfriet. 2012. *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. Encyclopedia of mathematics and its applications, Vol. 138. Cambridge University Press.

[14] Karel Culik II and Juhani Karhumäki. 1986. The equivalence of finite valued transducers (on HDT0L languages) is decidable. *Theor. Comput. Sci.* 47 (1986), 71–84.

[15] Luc Dartois, Paulin Fournier, Ismaël Jecker, and Nathan Lhote. 2017. On Reversible Transducers. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland (LIPIcs, Vol. 80)*, Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 113:1–113:12.

[16] Luc Dartois, Ismaël Jecker, and Pierre-Alain Reynier. 2018. Aperiodic String Transducers. *Int. J. Found. Comput. Sci.* 29, 5 (2018), 801–824. https://doi.org/10.1142/S0129054118420054

[17] Rodrigo De Souza. 2008. *Etude structurelle des transducteurs de norme bornée*. Ph. D. Dissertation. LTCI - Laboratoire Traitement et Communication de l'Information, Paris-Saclay. http://www.theses.fr/2008ENST0023/document

[18] Gaëtan Douéneau-Tabot. 2023. *Optimization of string transducers*. Ph. D. Dissertation. Université Paris Cité, Paris, France. https://gdoueneau.github.io/pages/DOUENEAU-TABOT_Optimization-transducers_v2.pdf

[19] Joost Engelfriet and Hendrik Jan Hoogeboom. 2001. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.* 2, 2 (2001), 216–254.

[20] Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. 2023. A Regular and Complete Notion of Delay for Streaming String Transducers. In *40th International Symposium on Theoretical Aspects of Computer Science, STACS 2023, March 7-9, 2023, Hamburg, Germany (LIPIcs, Vol. 254)*, Petra Berenbrink, Patricia Bouyer, Anuj Dawar, and Mamadou Moustapha Kanté (Eds.). Schloss

Dagstuhl - Leibniz-Zentrum für Informatik, 32:1–32:16. https://doi.org/10.4230/LIPIcs.STACS.2023.32

[21] Emmanuel Filiot and Pierre-Alain Reynier. 2021. Copyful Streaming String Transducers. *Fundam. Informaticae* 178, 1-2 (2021), 59–76. https://doi.org/10.3233/FI-2021-1998

[22] Patrick C. Fischer and Arnold L. Rosenberg. 1968. Multi-tape one-way nonwriting automata. *J. Comput. and System Sci.* 2 (1968), 88–101.

[23] Victor S. Guba. 1986. Equivalence of infinite systems of equations in free groups and semigroups to finite subsystems. *Mat. Zametki* 40, 3 (1986), 688—690.

[24] Eitan M. Gurari and Oscar H. Ibarra. 1981. The Complexity of Decision Problems for Finite-Turn Multicounter Machines. *J. Comput. Syst. Sci.* 22, 2 (1981), 220–229. https://doi.org/10.1016/0022-0000(81)90028-3

[25] Eitan M. Gurari and Oscar H. Ibarra. 1981. The Complexity of Decision Problems for Finite-Turn Multicounter Machines. In *Automata, Languages and Programming, 8th Colloquium, Acre (Akko), Israel, July 13-17, 1981, Proceedings (Lecture Notes in Computer Science, Vol. 115)*, Shimon Even and Oded Kariv (Eds.). Springer, 495–505.

[26] Eitan M. Gurari and Oscar H. Ibarra. 1983. A note on finite-valued and finitely ambiguous transducers. *Math. Syst. Theory* 16, 1 (1983), 61–66.

[27] Bernard R. Hodgson. 1983. Décidabilité par automate fini. *Ann. Sci. Math. Québec* 7, 3 (1983), 39–57.

[28] Oscar H. Ibarra. 1978. The unsolvability of the equivalence problem for e-free NGSM's with unary input (output) alphabet and applications. *SIAM J. of Comput.* 7, 4 (1978), 524–532.

[29] Gérard Jacob. 1977. Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. *Theoretical Computer Science* 5, 2 (1977), 183–204.

[30] Ismaël Jecker. 2021. A Ramsey Theorem for Finite Monoids. In *38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021, March 16-19, 2021, Saarbrücken, Germany (Virtual Conference) (LIPIcs, Vol. 187)*, Markus Bläser and Benjamin Monmege (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 44:1–44:13. https://doi.org/10.4230/LIPICS.STACS.2021.44

[31] J. Howard Johnson. 1985. Do Rational Equivalence Relations have Regular Cross-Sections?. In *Automata, Languages and Programming, 12th Colloquium, Nafplion, Greece, July 15-19, 1985, Proceedings (Lecture Notes in Computer Science, Vol. 194)*, Wilfried Brauer (Ed.). Springer, 300–309. https://doi.org/10.1007/BFB0015755

[32] Bakhadyr Khoussainov and Anil Nerode. 1995. Automatic Presentations of Structures. In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994 (Lecture Notes in Computer Science, Vol. 960)*. Springer, 367–392.

[33] Jérémy Ledent. 2013. Streaming string transducers (internship report). https://perso.ens-lyon.fr/jeremy.ledent/internship_report_L3.pdf.

[34] Arnaldo Mandel and Imre Simon. 1977. On finite semigroups of matrices. *Theoretical Computer Science* 5, 2 (1977), 101–111.

[35] Anca Muscholl and Gabriele Puppis. 2019. Equivalence of Finite-Valued Streaming String Transducers Is Decidable. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece (LIPIcs, Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 122:1–122:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.122

[36] Brigitte Rozoy. 1986. Outils et résultats pour les transducteurs boustrophédons. *RAIRO-Theoretical Informatics and Applications* 20, 3 (1986), 221–250.

[37] Aleksi Saarela. 2015. Systems of word equations, polynomials and linear algebra: A new approach. *Eur. J. Comb.* 47 (2015), 1–14. https://doi.org/10.1016/j.ejc.2015.01.005

[38] Jacques Sakarovitch. 1998. A Construction on Finite Automata that has Remained Hidden. *Theor. Comput. Sci.* 204, 1-2 (1998), 205–231. https://doi.org/10.1016/S0304-3975(98)00040-1

[39] Jacques Sakarovitch and Rodrigo de Souza. 2008. On the Decidability of Bounded Valuedness for Transducers. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings (Lecture Notes in Computer Science, Vol. 5162)*, Edward Ochmanski and Jerzy Tyszkiewicz (Eds.). Springer, 588–600.

[40] Jacques Sakarovitch and Rodrigo de Souza. 2008. On the decomposition of k-valued rational relations. In *STACS 2008, 25th Annual Symposium on Theoretical Aspects of Computer Science, Bordeaux, France, February 21-23, 2008, Proceedings (LIPIcs, Vol. 1)*, Susanne Albers and Pascal Weil (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 621–632. https://doi.org/10.4230/LIPICS.STACS.2008.1324

[41] Helmut Seidl. 1994. Equivalence of Finite-Valued Tree Transducers Is Decidable. *Math. Syst. Theory* 27, 4 (1994), 285–346. https://doi.org/10.1007/BF01192143

[42] John C. Shepherdson. 1959. The Reduction of Two-Way Automata to One-Way Automata. *IBM J. Res. Dev.* 3, 2 (1959), 198–200. https://doi.org/10.1147/RD.32.0198

[43] Andreas Weber. 1993. Decomposing Finite-Valued Transducers and Deciding Their Equivalence. *SIAM J. Comput.* 22, 1 (1993), 175–202. https://doi.org/10.1137/0222014

[44] Andreas Weber. 1996. Decomposing A k-Valued Transducer into k Unambiguous Ones. *RAIRO-ITA* 30, 5 (1996), 379–413.

[45] Andreas Weber and Helmut Seidl. 1991. On the Degree of Ambiguity of Finite Automata. *Theor. Comput. Sci.* 88, 2 (1991), 325–349. https://doi.org/10.1016/0304-3975(91)90381-B

[46] Di-De Yen and Hsu-Chun Yen. 2022. On the decidability of the valuedness problem for two-way finite transducers. *Inf. Comput.* 285, Part (2022), 104870. https://doi.org/10.1016/J.IC.2022.104870